Lossy Trapdoor Functions and Their Applications*

Chris Peikert†
Georgia Institute of Technology

Brent Waters‡
University of Texas at Austin

June 4, 2010

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 based on a variety of cryptographic assumptions, including the hardness of the decisional Diffie-Hellman (DDH) problem, and the hardness of the "learning with errors" problem (which is implied by the *worst-case* hardness of various lattice problems).

Taken together, our results resolve some long-standing open problems in cryptography. They give the first injective trapdoor functions based on problems not directly related to integer factorization, and provide the first 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).

†A majority of this work was performed while at SRI International. 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.

‡A majority of this work was performed while at SRI International. 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; and 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) [47, 55, 23]. 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 [21] and were first realized in the RSA function of Rivest, Shamir, and Adelman [58]. 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.

Known constructions of TDFs all rely upon the particular algebraic properties of the functions. For CCA security, the main construction paradigm in the existing literature relies upon *noninteractive zero-knowledge* (NIZK) proofs [10, 26] (either for general NP statements or for specific number-theoretic problems). Such proofs allow the decryption algorithm to check that a ciphertext is 'well-formed,' and (informally speaking) force the adversary to produce only ciphertexts for which it already knows the underlying messages, making its decryption oracle useless.

Unfortunately, it is still not known how to realize TDFs and CCA security (in the standard model) based on all the types of assumptions described above. Using NIZK proofs, CCA-secure cryptosystems have been constructed based on problems related to factoring and discrete logs [47, 23, 60, 19, 20], 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 [58, 54, 48].

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 well-studied 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 conceptually simple *black-box* construction of a CCA-secure cryptosystem based on lossy TDFs. In contrast to prior approaches, the decryption algorithm in our scheme is *witness-recovering*, i.e., along with the message it also recovers the *randomness* that was used to create the ciphertext. It then checks well-formedness simply by re-encrypting the message under the retrieved randomness, and comparing the result to the original ciphertext. Until now, witness-recovering CCA-secure cryptosystems were known to exist only in the random oracle model [8, 28].

  Our approach has two main benefits: first, the cryptosystem uses its underlying primitive (lossy TDFs) as a "black-box," making it more efficient and technically simpler than those that follow the general

NIZK paradigm [47, 23, 60].[1] Second, it yields the first known CCA-secure cryptosystem based entirely on (worst-case) lattice assumptions, resolving a problem that has remained open since the pioneering work of Ajtai [1] and Ajtai and Dwork [2].[2]

- 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 [34, 35], our OT protocols additionally imply general secure multiparty computation for malicious adversaries.

1.1 Trapdoor Functions and Witness-Recovering Decryption

Trapdoor functions are certainly a powerful and useful primitive in cryptography. Because they generically yield *passively secure* (i.e., chosen plaintext-secure) cryptosystems that are witness-recovering, it is tempting to think that they might also yield efficient CCA-secure encryption via witness recovery. Indeed, this approach has borne some fruit [6, 8, 28], but so far only with the aid of the random oracle heuristic.

A related long-standing question is whether it is possible to construct (a collection of) trapdoor functions from any cryptosystem that is secure under a chosen-plaintext attack (CPA-secure) [6]. A tempting approach is to generate the function description as a public encryption key pk, letting its trapdoor be the matching secret decryption key sk, and defining $f_{pk}(x) = E_{pk}(x; x)$. That is, encrypt the input x, also using x itself as the random coins for encryption (for simplicity we ignore the possibility that encryption may require more random bits than the message length). The cryptosystem's completeness ensures that decrypting the ciphertext with the secret key (i.e., the function's trapdoor) returns x. The only remaining question is whether this function is one-way, assuming that the cryptosystem is CPA-secure.

Unfortunately, we have no reason to think that the above function (or anything like it) is hard to invert, because CPA security is guaranteed only if the randomness is chosen *independently* of the encrypted message. For example, consider a (pathological, but CPA-secure) encryption algorithm E', which is built from another (CPA-secure) encryption algorithm E: the encryption algorithm $E'(m; r)$ normally returns $E(m; r)$, except if $m = r$ it simply outputs r. Then our candidate trapdoor function $f_{pk}(x) = E'(x; x)$ 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 [31] demonstrated a *black-box separation* between injective (or even poly-to-one) trapdoor functions and CPA-secure encryption. Intuitively, the main difference is that inverting a trapdoor function requires the recovery of its *entire* input, whereas a decryption algorithm only needs to recover the input message, but not necessarily the encryption randomness. For similar reasons, there is also some evidence that achieving CCA security from CPA security (in a black-box manner) would be difficult [30].

Perhaps for these reasons, constructions of CCA-secure encryption in the standard model [47, 23, 60, 19, 20] have followed a different approach. As explained in [24], all the techniques used so far have employed a "many-key" construction, where the well-formedness of a ciphertext is guaranteed by a (simulation-sound) non-interactive zero knowledge (NIZK) proof that the same message is encrypted under two or more public keys. A primary benefit of zero-knowledge is that the decryption algorithm can ensure 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 [23], as well as efficient systems based on specific number-theoretic problems [19, 20], such as

[1] We note that Cramer and Shoup [19, 20] gave *efficient* CCA-secure constructions based on NIZK proofs for *specific* number-theoretic problems.

[2] We also note that while NIZK proofs for certain lattice problems are known [51], they do not appear to suffice for CCA security.

the decisional Diffie-Hellman (DDH) [13] and decisional composite residuosity [48] problems. However, the NIZK approach has two significant drawbacks. First, the constructions from general assumptions are *inefficient*, as they are inherently non-black-box and require NIZK proofs for general NP statements. Second, while CPA-secure public key cryptosystems based on *worst-case* lattice assumptions are known [2, 56, 57], there are still no known *CCA-secure* systems, because it is unknown how to realize NIZKs for all of NP (or even 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 (i.e., without random oracles). As discussed above, past experience suggests that we might need to build from a stronger base notion than chosen-plaintext security.

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 corresponds to 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., most outputs of f have many preimages. Finally, the two behaviors are *indistinguishable*: given just the public 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 *outputs* of the scheme (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 trapdoor for 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. An 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 function's random input 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' $k = n/2$ bits of the input. Then the *injective* functions from this collection make up a collection of one-way 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 $f(x)$, the inverter must output (with some noticeable probability) the same value x, because f is injective and by hypothesis on $\mathcal{I}$. Now consider running 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. (Formally, x has large *average min-entropy* given $f(x)$.) Therefore, even though a (potentially unbounded) inverter might find one of these several preimages, it cannot guess the *particular* preimage x that we have in mind (except with negligible probability). We conclude that no efficient inverter can exist for the injective functions, unless the injective and lossy functions are distinguishable via the above test.

Using the fact that lossy TDFs imply standard injective TDFs, we can construct a CPA-secure cryptosystem by standard techniques. For instance, a well-known folklore construction uses the generic Goldreich-Levin hard-core predicate [33] for $f(x)$ to conceal a message bit, and uses the trapdoor in decryption to invert f and recover the bit.

However, it is instructive (and a useful warm-up for our CCA-secure construction) to see that lossy TDFs admit many-bit *hard-core functions*, via a very simple and direct proof of security (which should be contrasted with the strong but complex machinery of the Goldreich-Levin theorem). Let $\mathcal{H}$ be a family of universal 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. Then a hash function h chosen at random from $\mathcal{H}$ is a hard-core function for the injective TDFs of the collection; i.e., $h(x) \in \{0,1\}^\ell$ is pseudorandom given $f(x)$ (and the description of h), where f is injective and $x \leftarrow \{0,1\}^n$ is uniformly random.

To see this, consider an adversary that attempts to distinguish $h(x) \in \{0,1\}^\ell$ from a truly uniform and independent string. A (computationally bounded) 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 $f'(x)$. By a suitable version of the leftover hash lemma [38, 22], h is a *strong randomness extractor*, so it follows that $h(x)$ is statistically close to uniform over $\{0,1\}^\ell$ given $f'(x)$ and h. Therefore, even an unbounded adversary has negligible distinguishing advantage in this experiment, and our claim is established.

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. In our construction above, if we 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 somewhat richer cryptographic abstraction called an *all-but-one* (ABO) trapdoor function, which can be constructed generically from a collection of sufficiently lossy TDFs, or more directly under the same 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. (Note that ABO trapdoor functions directly imply lossy trapdoor functions: simply include with the ABO function the value of a branch that is either injective or lossy.)

Cryptosystem. For simplicity, and because it captures the main novelty of our approach, here we describe a cryptosystem that is secure under a CCA1, or "lunchtime," attack. In such an attack, the adversary has access to a decryption oracle only *before* receiving its challenge ciphertext. A standard transformation can then immunize the basic scheme against full CCA2 attacks (see the remarks following the proof sketch).

Our construction uses a collection of lossy TDFs and a collection of ABO TDFs, both of which have domain $\{0,1\}^n$. Assume that the lossy functions from these collections individually lose (say) $k \geq (2/3)n$ of their input bits, so that when evaluated on the same input they still jointly lose at least $n - 2(n-k) \geq n/3$ bits. As in our CPA-secure scheme above, it also uses a universal family of hash functions $\mathcal{H}$ from $\{0,1\}^n$ to

$\{0,1\}^\ell$, where $\ell \approx n/3$ is the length of the plaintext messages, determined by the joint lossiness. (We defer the exact selection of parameters to Section 4.)

As in the basic CPA-scheme, the cryptosystem's key generator generates an injective function f from the lossy TDF collection, along with its trapdoor f^{-1}, and chooses a hash function h at random from $\mathcal{H}$. In addition, it generates an ABO function g whose lossy branch $b^* \in B$ is uniformly random (decryption will not require the trapdoor for g, so it may be discarded). 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 chooses an input $x \in \{0,1\}^n$ and a branch $b \in B$ uniformly at random. The ciphertext is

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

We emphasize that both f and g are evaluated on the same input x.

The decryption algorithm attempts to decrypt a ciphertext $c = (b, c_1, c_2, c_3)$ as follows: it computes some $x' = f^{-1}(c_1)$ using its trapdoor for f, obtaining a *witness* x'. Then it simply recomputes the ciphertext to verify that it is well-formed, by checking that $c_1 = f(x')$ and $c_2 = g(b, x')$, and aborting if not. Finally, it outputs the message $m' = h(x') \oplus c_3$.

Sketch of security proof. The proof of security follows a hybrid two-key argument, but without zero knowledge (due to the recovery of the encryption witness). The proof involves a few hybrid experiments that are indistinguishable to any efficient adversary. In the first hybrid game, the challenge ciphertext is computed using the lossy branch $b = b^*$ of the ABO function g; this game is indistinguishable from the real attack because the description of g hides b^*. In the next hybrid game, the decryption oracle extracts its witness x' using a trapdoor g^{-1} for the ABO function, rather than with f^{-1} (the validity test remains the same). Because there is at most one witness x' that satisfies the validity test for any ciphertext, the modified decryption oracle behaves exactly as the real one does for *all branches but one*, namely, the b^* branch — but again the adversary cannot issue any query on branch b^* because it is hidden. (Of course, once the adversary receives its challenge ciphertext, the lossy branch b^* is revealed, which is why this scheme's security is limited to lunchtime attacks.) In the final hybrid game, the injective function f is replaced with a lossy one. At this point, we observe that the two components $c_1 = f(x)$ and $c_2 = g(b^*, x)$ of the challenge ciphertext jointly lose about ℓ bits of x. Therefore, $h(x)$ is statistically close to uniform (given the rest of the view of the adversary), and even an unbounded adversary has only negligible advantage in this final game. It follows that an efficient adversary's advantage is also negligible in the real attack.

Remarks. We conclude this summary with a few notes. First, in practice one would likely use our construction as a public-key key encapsulation mechanism (KEM), where the encapsulated key is simply $h(x)$. (A KEM does not encrypt messages directly; rather, it simply allows an encrypter and decrypter to agree on a shared key that appears random to the attacker.)

Second, for a full CCA2 attack, the only remaining requirement is non-malleability of the challenge ciphertext (ciphertexts in the above scheme are clearly malleable by just flipping bits of $c_3 = m \oplus h(x)$). In our CCA2-secure scheme, we employ the technique due to Dolev, Dwork, and Naor [23] of signing each ciphertext with a fresh key from a strongly unforgeable one-time signature scheme, and we use the public verification key (or a hash of it) as the branch of the ABO function.[3] The unforgeability property ensures that the attacker cannot make a valid query on the lossy branch, even after learning its value. Note that depending

[3] Strongly unforgeable one-time signatures are implied by one-way functions, and in particular by lossy trapdoor functions.

on the implementation of the signatures scheme, the resulting decryption algorithm might no longer recover the *entire* encryption witness (which now includes randomness for generating a key pair for the signature scheme), but it still recovers the input x of the lossy and ABO functions f, g.

Finally, while our system falls outside the NIZK paradigm, we do rely on some techniques that are reminiscent of previous CCA-secure constructions. Our construction uses a two-key strategy originally due to Naor and Yung [47], 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, whereas our simulation participates in a distinguishing experiment on the *other key itself*.

1.3 Realizing Lossy TDFs

We now sketch our basic approach for constructing lossy and all-but-one trapdoor functions. The starting point is to rely on cryptographic assumptions that are amenable to linear (homomorphic) operations on encrypted data. Informally, this means that one can apply linear operations (i.e., addition, multiplication by a known scalar) directly on encrypted values, without first decrypting them.

A function f (whether injective or lossy) on $\{0,1\}^n$ is specified by an $n \times n$ matrix $\mathbf{M}$ that has been encrypted in a suitable way. The underlying hardness assumption ensures that any two encrypted matrices are computationally indistinguishable (for any n bounded by a polynomial in the security parameter), hence so are injective and lossy functions.

To evaluate $f(x)$, we view the input $x \in \{0,1\}^n$ as an n-dimensional binary vector $\mathbf{x}$, and simply compute the encrypted linear product $\mathbf{x} \cdot \mathbf{M}$ by applying the appropriate homomorphic operations to the encrypted entries of $\mathbf{M}$.

An *injective* function is represented by an encrypted *identity* matrix $\mathbf{M} = \mathbf{I}$, and its trapdoor is the matching decryption key. Briefly, the function f is invertible with the trapdoor (and therefore injective) 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 individually.

A *lossy* function is represented by an encrypted *all-zeros* matrix $\mathbf{M} = \mathbf{0}$. Because $\mathbf{x} \cdot \mathbf{0} = \mathbf{0}$ for all $\mathbf{x}$, the function value $f(x)$ intuitively 'loses' x, at least from the point of view of an *honest* party who might try to decrypt the output. However, this alone is not enough to guarantee statistical lossiness, because the output ciphertexts still carry some internal *randomness* that might leak information about the input.

To ensure lossiness, we rely on a special method of encrypting the matrix $\mathbf{M}$. With this method, every row of ciphertexts in an encrypted zero matrix lies in a *low-dimensional subspace* of the n-fold ciphertext space. (Despite this fact, the plaintext matrix is still concealed.) The point of this property is to ensure that every homomorphically computed linear combination of the rows (i.e., every value of $f(x)$) also lies in the same subspace. By choosing the dimension n so that the number of inputs 2^n significantly exceeds the cardinality of the subspace, we can then ensure that the image of f is much smaller than its domain, as desired.

The construction of all-but-one trapdoor functions is just slightly more general. Each branch b of the function simply corresponds to a different matrix $\mathbf{M}_b$, whose encrypted version can be derived easily from the function's description. The function is generated so that $\mathbf{M}_{b^*} = \mathbf{0}$ for the desired lossy branch b^*, while $\mathbf{M}_b$ is an invertible matrix (which is easily computable with the trapdoor) for all the other branches.

We note that it is tempting to think that the above approach could be made more space-efficient by using an encrypted n-dimensional *vector* as the function description, and component-wise multiplication for evaluation. While this presents no problem when inverting an injective function, it does not suffice for lossiness (at least generically). The problem is that the encrypted output vector does not necessarily lie within a particular subspace, so the number of outputs cannot be bounded appropriately.

Concrete assumptions. Using well-known techniques relating to the decisional Diffie-Hellman problem, it is relatively straightforward to implement the above framework for constructing lossy and all-but-one TDFs under the DDH assumption. One subtlety is that in order to support the needed additive homomorphisms, encrypted values must appear 'in the exponent,' so decryption would in general require computing discrete logarithms. Fortunately, in our context all the encrypted values are guaranteed to be binary, so decryption is easy. (See Section 5 for details).

Instantiating lossy TDFs under worst-case lattice assumptions via known lattice-based cryptosystems [2, 56, 57] encounters some additional technical difficulties. While all these cryptosystems have a linear structure that supports additive homomorphisms, their security (roughly speaking) is based on the difficulty of distinguishing uniform points in an n-dimensional space from those that are 'close to' a subspace of dimension $n - 1$. In our matrix framework, the lossiness argument encounters two significant problems: first, the outputs of a lossy function are only guaranteed to lie 'somewhat close' to the subspace, and the subspace itself may already have cardinality much larger than the number of function inputs. We address these problems first by 'lifting' the underlying problem to that of distinguishing between uniform points and points that are close to a subspace of *much smaller* dimension; and second, by fitting the function output itself into a *low-dimensional space* to control the number of possible outputs in the lossy case (while retaining the ability to invert in the injective case).

Technically, the techniques described above (especially the lifting step) seem easiest to implement using the natural "learning with errors" (LWE) problem as defined by Regev [57], which is a generalization of the well-known "learning parity with noise" problem to moduli larger than 2.[4] The LWE problem can be seen as an average-case bounded-distance decoding problem on a certain natural family of random lattices, and appears to be quite hard (the best known attack [9] requires exponential time and space). Moreover, Regev gave a reduction showing that LWE is indeed hard on the average if standard approximation problems on lattices are hard in the *worst case* for *quantum* algorithms [57]. Quantum algorithms are not known to have any advantage over classical algorithms for the worst-case lattice problems in question. In addition, following the initial publication of this work, Peikert [50] has shown that LWE is as hard as certain worst-case lattice problems via a *classical* reduction.

1.4 Lossy Trapdoors in Context

It is informative to consider lossy trapdoors in the context of previous constructions. A crucial technique in the use of lossy trapdoors is that security is typically demonstrated via indistinguishability arguments over a scheme's *public key*, as opposed to its outputs. For encryption, this style of argument goes back the seminal work of Goldwasser and Micali [36], and recently has been identified as an important notion (called "message-lossy" [52] or "meaningful/meaningless" [42] encryption) in its own right. The style is inherent to cryptosystems based on lattices [2, 56, 57], but to our knowledge it has never been employed in the context of trapdoor functions or chosen-ciphertext security.

The present approach can be contrasted with the (1-out-of-2) oblivious transfer (OT) construction of Even, Goldreich, and Lempel [25]. They construct semi-honest oblivious transfer protocols from any public key cryptosystem in which a public key can be sampled 'obliviously,' i.e., without knowing a corresponding decryption key. 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*

[4] Concurrently with the initial version of this work, Ajtai and Dwork [3] improved their original cryptosystem to include a lifting argument that also appears amenable to our framework.

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) [63] that were introduced by Canetti, Halevi, and Katz [17] and improved in later work [15, 16, 14]. 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, which generate a matrix whose rows lie in a small subspace, are technically similar to the ElGamal-like cryptosystems of Bellare *et al.* [5] that reuse randomness for efficiency, and to constructions of pseudorandom functions (via intermediate objects called "synthesizers") by Naor and Reingold [45]. 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

Since the initial publication of this work in [53], there has been much additional work on lossy trapdoor functions and related concepts.

Additional constructions and variations. One area of interest has been in finding additional realizations of lossy trapdoor functions. Rosen and Segev [59] and Boldyreva, Fehr, and O'Neill [12] independently described simple, compact constructions of lossy and ABO TDFs under the decisional composite residuosity assumption, using the trapdoor function of Paillier [48]. (The preliminary version of this work [53] constructed somewhat more complex lossy and ABO TDFs under a variant of Paillier's assumption.) More recently, Freeman, Goldreich, Kiltz, Rosen and Segev [27] produced more constructions of lossy TDFs, from the quadratic residuosity assumption and the family of k-linear assumptions [39, 62] (which are potentially weaker generalizations of the DDH assumption). Boyen and Waters gave a technique to 'compress' the public key of our matrix construction down to $O(n)$ group elements in a 'pairing-friendly' group.

Another direction of research has been to give further applications and variations of lossiness. Boldyreva *et al.* [12] constructed CCA-secure *deterministic* encryption schemes for high-entropy messages, based on lossy and ABO TDFs. Peikert, Vaikuntanathan, and Waters [52] constructed efficient, universally composable oblivious transfer protocols based on certain "message-lossy" encryption schemes, and Bellare, Hofheinz, and Yilek [7] proved that message-lossy encryption schemes are secure under "selective-opening attacks." Rosen and Segev [59] introduced a relaxation of lossiness, called security under "correlated inputs," and constructed a witness-recovering CCA-secure cryptosystem using that notion. Mol and Yilek [44] recently solved an open problem from an earlier version of this work, by constructing a CCA-secure encryption scheme from any lossy TDF that loses only a noticeable fraction of a bit.

Trapdoors for lattices. Using very different techniques from ours, Gentry, Peikert, and Vaikuntanathan [29] recently constructed two different types of trapdoor functions that are secure under worst-case lattice assumptions. One collection consists of injective functions that can be shown secure under correlated

inputs [59] (they also can be modified to enjoy a mild degree of lossiness). Peikert [50] has adapted this collection to construct a simple CCA-secure cryptosystem based directly on the LWE assumption.

1.6 Open Directions

This work leaves a number of open problems and directions, some of which we discuss briefly here.

- Are there constructions of lossy trapdoor functions (or even just lossy functions, without trapdoors) based on other assumptions? All known constructions are based on specific algebraic problems; a construction based on 'general' cryptographic assumptions would be quite interesting. A related question is whether more efficient constructions based on the DDH and LWE problems are possible.
- Is it possible to 'amplify' lossiness in an unconditional way? That is, can one construct a 'highly lossy' function that loses a δ fraction of its input, from a 'moderately lossy' one that loses only an $\epsilon < \delta$ fraction?
- What other cryptographic applications do lossy TDFs imply? For example, (single-server) private information retrieval (PIR) is an 'information-compressing' primitive, but we do not know of a construction from lossy TDFs (or vice-versa).

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). The security parameter, represented in unary, is an input to all cryptographic algorithms (including the adversary); for notational clarity we usually omit it as an explicit parameter.

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 $\mathsf{poly}(\lambda)$ 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 $\mathsf{poly}(\lambda)$..

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*, written $\mathcal{X} \overset{s}{\approx} \mathcal{Y}$, if $\Delta(X_\lambda, Y_\lambda) = \mathsf{negl}(\lambda)$. 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*, written $\mathcal{X} \overset{c}{\approx} \mathcal{Y}$, if the advantage of any PPT algorithm $\mathcal{A}$ is $\mathsf{negl}(\lambda)$.[5] It is routine to see that statistical indistinguishability implies computational indistinguishability. When the ensembles of two random variables (indexed by λ) are clear from context, we sometimes abuse notation and say that the variables themselves are statistically/computationally indistinguishable.

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 $\mathcal{X}_1 \overset{s}{\approx} \mathcal{X}_2 \overset{s}{\approx} \cdots \overset{s}{\approx} \mathcal{X}_k$ (respectively, $\mathcal{X}_1 \overset{c}{\approx} \cdots \overset{c}{\approx} \mathcal{X}_k$) is any sequence of $k = \text{poly}(\lambda)$ ensembles, then $\mathcal{X}_1 \overset{s}{\approx} \mathcal{X}_k$ (resp., $\mathcal{X}_1 \overset{c}{\approx} \mathcal{X}_k$).

2.2 Cryptographic Notions

Here we recall some standard cryptographic definitions. We frequently define security 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.

2.2.1 Trapdoor Functions

We recall one definition 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 $f_s^{-1}(\cdot)$. (Outside the image of f_s, the behavior of $F^{-1}(t, \cdot)$ may be arbitrary.)

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.

2.2.2 Collision-Resistant and One-Way Hash Functions

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

[5] 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.

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) [46] 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 $x' \in \{0,1\}^{\ell(\lambda)}$. Then the probability (over all the randomness of the game) that $x' \neq x$ and $F_{\text{uowhf}}(i, x) = F_{\text{uowhf}}(i, x')$ is $\mathsf{negl}(\lambda)$. It is easy to see that a collection of CRHFs is also a collection of UOWHFs.

2.2.3 Public-Key Encryption

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 experiment.

A basic notion of security for a public key cryptosystem is indistinguishability under a chosen plaintext attack, called *CPA security* (also sometimes referred to as *semantic 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).

2.2.4 Strongly Unforgeable One-Time Signatures

A signature scheme 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 Ver accepts (i.e., outputs 1) with overwhelming probability over all the randomness of the experiment.

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 $\mathcal{A}$ may query an oracle that computes $\mathsf{Sign}(sk_\sigma, \cdot)$ on a single message $m \in \mathcal{M}$ of its choice, receiving a signature σ. Finally, $\mathcal{A}$ outputs a pair (m', σ'), and is said to succeed if $\mathsf{Ver}(vk, m', \sigma') = 1$ and, if a signature query was made, $(m', \sigma') \neq (m, \sigma)$. 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 [32, Chapter 6], and more efficiently from collision-resistant hash functions [41]. As we show later, both of these primitives have black-box constructions from lossy trapdoor functions.

2.3 Randomness Extraction

The min-entropy of a random variable X over a domain S is the negative (base-2) logarithm of the *predictability* of X:

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

In many natural settings (including our applications), the variable X is correlated with another variable Y whose value is known to an adversary. In such scenarios, it is most convenient to use the notion of *average min-entropy* as defined by Dodis *et al.* [22], which captures the *average predictability* of X conditioned on the (random) value of Y:

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

(See [22] for further discussion and alternate notions.)

Lemma 2.1 ([22, Lemma 2.2])**.** *If Y takes at most 2^r possible values and Z is any random variable, then*

$$\tilde{H}_\infty(X|Y) \geq H_\infty(X) - r.$$

In our applications, we need to derive nearly uniform bits from a weakly random source X that has some average min-entropy. This is accomplished via an appropriate type of *randomness extractor*.

Definition 2.2 ([22]). *A collection $\mathcal{H}$ of functions from $\{0,1\}^n$ to $\{0,1\}^\ell$ is an* average-case (n, k, ℓ, ϵ)-strong extractor *if for all pairs of random variables (X, Y) such that $X \in \{0,1\}^n$ and $\tilde{H}_\infty(X|Y) \geq k$, it holds that for $h \leftarrow \mathcal{H}$ and $r \leftarrow \{0,1\}^\ell$,*

$$\Delta((h, h(X), Y), (h, r, Y)) \leq \epsilon.$$

Dodis *et al.* [22] showed that in general, any (worst-case) strong extractor is also an average-case strong extractor, for an appropriate setting of parameters. However, the proof incurs some degradation in the number of bits that may be extracted (and their quality) for a given amount of average min-entropy. We instead prefer to use a tighter but more specialized result, which says that universal hash functions are good average-case strong extractors. (Recall that a family of functions $\mathcal{H} = \{h_i : D \to R\}$ from a domain D to range R is said to be *universal* if, for every distinct $x, x' \in D$, $\Pr_{h \leftarrow \mathcal{H}}[h(x) = h(x')] = 1/|R|$. Universal hash functions admit very simple and efficient constructions [64].)

Lemma 2.3 ([22, 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 universal hash functions from $\{0,1\}^n$ to $\{0,1\}^\ell$, where $\ell \leq k - 2\lg(1/\epsilon)$. Then $\mathcal{H}$ is an average-case (n, k, ℓ, ϵ)-strong extractor.*

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 $(S_{\text{inj}}, S_{\text{loss}}, F_{\text{ltdf}}, F_{\text{ltdf}}^{-1})$ having the properties below.

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 $F_{\text{ltdf}}^{-1}(t, \cdot)$ 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 $F_{\text{ltdf}}^{-1}(t, y)$ is unspecified (because of this, the output of F_{ltdf}^{-1} 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_{ltdf}^{-1} 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 $F_{\text{ltdf}}^{-1}(t, \cdot)$ 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:* the views of any PPT adversary $\mathcal{A}$ in the following two experiments, indexed by a bit $i \in \{0,1\}$, are computationally indistinguishable: $\mathcal{A}$ outputs $(b_0^*, b_1^*) \in B_\lambda \times B_\lambda$ and is given a function index s, where $(s, t) \leftarrow S_{\text{abo}}(b_i^*)$.

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. As with lossy TDFs, the use of almost-always ABOs 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. *There exists a collection of (n, k)-ABO TDFs having exactly two branches if and only if there exists a collection of (n, k)-lossy TDFs.*

Proof. Suppose that $(S_{\text{abo}}, G_{\text{abo}}, G_{\text{abo}}^{-1})$ give an (n, k)-ABO collection having branch set $\{0,1\}$ (without loss of generality). We construct $(S_{\text{inj}}, S_{\text{loss}}, F_{\text{ltdf}}, F_{\text{ltdf}}^{-1})$ that give a collection of (n, k)-lossy TDFs as follows:

- The generator S_{inj} outputs $(s, t) \leftarrow S_{\text{abo}}(1)$, and S_{loss} outputs $(s, \perp)$ where $(s, t) \leftarrow S_{\text{abo}}(0)$.

- The evaluation algorithm F_{ltdf} always evaluates on branch $b = 0$, i.e., $F_{\text{ltdf}}(s, x) = G_{\text{abo}}(s, 0, x)$.
- The inversion algorithm $F^{-1}_{\text{ltdf}}(t, y)$ outputs $x \leftarrow G^{-1}_{\text{abo}}(t, 0, y)$.

It is clear by construction that F^{-1}_{ltdf} correctly inverts (using the trapdoor) any function generated by S_{inj} because the underlying ABO function is evaluated on a non-lossy branch, whereas S_{loss} generates a lossy function having image size at most 2^{n-k} because the underlying ABO is evaluated on its lossy branch. Moreover, injective and lossy functions are computationally indistinguishable by the hidden lossy branch property of the ABO.

Now consider the converse direction, supposing that $(S_{\text{inj}}, S_{\text{loss}}, F_{\text{ltdf}}, F^{-1}_{\text{ltdf}})$ give a collection of (n, k)-lossy TDFs. We construct $(S_{\text{abo}}, G_{\text{abo}}, G^{-1}_{\text{abo}})$ that give an (n, k)-ABO collection having branch set $B = \{0, 1\}$ as follows:

- The generator $S_{\text{abo}}(b^*)$ chooses $(s'_0, \perp) \leftarrow S_{\text{loss}}$, $(s'_1, t) \leftarrow S_{\text{inj}}$, and outputs $(s = (s'_{b^*}, s'_{1-b^*}), t)$.
- The evaluation algorithm $G_{\text{abo}}(s = (s_0, s_1), b, x)$ outputs $F_{\text{ltdf}}(s_b, x)$.
- The inversion algorithm $G^{-1}_{\text{abo}}(t, b, y)$ outputs $F^{-1}_{\text{ltdf}}(t, y)$.

Using the properties of the lossy TDF collection, it may be verified that the function computed by G_{abo} is lossy on branch b^*, and injective (and invertible by G^{-1}_{abo}) on branch $1 - b^*$. Finally, for either input $b^* \in \{0, 1\}$ to S_{abo}, the output function index s is computationally indistinguishable from one made up of two indices independently generated by S_{inj}, hence the ABO satisfies the hidden lossy branch condition. □

We can also construct an ABO collection for large 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. *If there exists an $(n, n - r)$-ABO collection with branch set $B = \{0, 1\}$, then for any $\ell \geq 1$ there exists an $(n, n - \ell \cdot r)$-ABO collection with branch set $B = \{0, 1\}^\ell$.*

Proof. Suppose by hypothesis that (S, G, G^{-1}) gives an $(n, n - r)$-ABO collection with branch set $\{0, 1\}$. We construct $(S_{\text{abo}}, G_{\text{abo}}, G^{-1}_{\text{abo}})$ that give an $(n, n - \ell \cdot r)$-ABO collection with branch set $\{0, 1\}^\ell$.

- $S_{\text{abo}}(b^*)$ generates ℓ individual functions $(s_i, t_i) \leftarrow S(b^*_i)$ for $i \in [\ell]$, where b^*_i is the ith bit of b^*. The output is $(s = (s_1, \ldots, s_\ell), t = (b^*, t_1, \ldots, t_\ell))$.
- $G_{\text{abo}}(s, b, x)$ computes $y_i = G(s_i, b_i, x)$ for each $i \in [\ell]$ and outputs $y = (y_1, \ldots, y_\ell)$.
- $G^{-1}_{\text{abo}}(t, b, y)$ finds an index $i \in [\ell]$ such that $b_i \neq b^*_i$, and outputs $x \leftarrow G^{-1}(t_i, b_i, y_i)$. (If $b = b^*$, G^{-1}_{abo} outputs $\perp$.)

One may check that G^{-1}_{abo} inverts correctly on any branch $b \neq b^*$, because y_i was computed on an injective branch when $b_i \neq b^*_i$. To analyze the leakage of $G_{\text{abo}}(s, b^*, \cdot)$, note that all ℓ subfunctions $G(s_i, b_i, \cdot)$ are evaluated on their lossy branches, so by hypothesis the total number of possible output values is at most $(2^r)^\ell = 2^{\ell \cdot r}$. Finally, the hidden lossy branch property follows by a routine hybrid argument over the components of the function index s. □

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 the construction of standard trapdoor functions). It seems plausible that realizing this weaker notion of "lossy (non-trapdoor) functions" could be achieved more simply or efficiently than the full notion 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 $(S_{ltdf}, F_{ltdf}, F^{-1}_{ltdf})$ give a collection of (n, k)-lossy trapdoor functions with $k \geq \omega(\log \lambda)$. Then $(S_{inj}, F_{ltdf}, F^{-1}_{ltdf})$ give a collection of injective trapdoor functions. (The analogous result applies for almost-always collections.)*

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

Suppose by way of contradiction that $\mathcal{I}$ is a PPT inverter for the collection, 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 1 ("injective"), otherwise output 0 ("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, let s be any fixed function index generated by S_{loss}. Then the probability (over the choice of x) that even an unbounded $\mathcal{I}$ predicts x is given by the average min-entropy of x conditioned on $f_s(x)$, i.e., the predictability of x given $f_s(x)$ is at most $2^{-\tilde{H}_\infty(x|f_s(x))}$. Because $f_s(\cdot)$ takes at most 2^{n-k} values, Lemma 2.1 implies that

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

Because $k = \omega(\lg \lambda)$, the probability that $\mathcal{I}(s, y)$ outputs x, and $\mathcal{D}$ outputs "injective," is $\mathsf{negl}(\lambda)$. By averaging, the same is true for s chosen at random by S_{loss}. We conclude that $\mathcal{D}$ distinguishes injective functions from lossy ones with nonnegligible advantage, a contradiction of the hypothesis. □

Note that in general, the hypothesis that $k = \omega(\log \lambda)$ seems necessary for showing (strong) one-wayness. The reason is that a lossy function may be injective on, say, a subset of size 2^{n-k-1} of the input domain $\{0,1\}^n$, and $(2^{k+1} - 1)$-to-1 on the remainder (one may check that such a function has image size 2^{n-k}). On this type of function, an unbounded inverter could choose to invert only those values having unique preimages, which occur with probability 2^{-k-1}, and to invert an injective function with the same probability. However, for *weak* one-wayness (where an inverter must succeed with probability negligibly close to 1), the proof of Lemma 3.3 is easily adapted to show that even $k = 1/\text{poly}(\lambda)$ suffices.

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 $h : \{0,1\}^n \to \{0,1\}^\ell$ such that $h(x)$ is computationally indistinguishable from a uniformly random $r \in \{0,1\}^\ell$, given the value $f(x)$.

Our results here can be contrasted with that of Goldreich and Levin [33], 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 sophisticated reduction whose running time depends on the distinguishing advantage of the distinguisher; our proof is elementary and the security reduction is tight (i.e., the running time and distinguishing advantage for lossy and injective functions are essentially the same as those for the distinguisher between $h(x)$ and r).

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 universal family of 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 distributions that are generated by the experiments described below, which are implicitly indexed by the security parameter λ.

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

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

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

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

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, by Lemma 2.1 we have

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

Therefore, by Lemma 2.3, the hypothesis that $\ell \leq k - 2\lg(1/\epsilon)$, and by averaging over all choices of $s \leftarrow S_{\text{loss}}$, 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 [11, 65, 38]. In particular, Håstad *et al.* [38] 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 [38, Sections 4.3 and 4.6] for details. We observe that because universal 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 [38], 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) an injective function, and h is selected at random from a universal family of hash functions. (As we will see, the construction and proof work equally well regardless of whether H is defined using an injective or lossy f.) For an appropriate output length of the universal hash functions, H shrinks its input, and for a sufficient amount of lossiness, finding collisions 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" invocation of h. Now consider the function $H = h \circ f'$, where f' is sufficiently lossy. Then with overwhelming probability, the function h contains *no collisions*, either with 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" invocation of f'. We can therefore distinguish between injective and lossy functions by checking whether an adversarially generated collision of H occurs in its outer or inner component. 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 $n - k \leq n/2 - d$ for some $d = \omega(\log \lambda)$. (We will not need the inversion algorithm F_{ltdf}^{-1}, and an almost-always collection of lossy functions also suffices.) Let $\mathcal{H} = \{h_i : \mathcal{R} \to \{0,1\}^\ell\}$ be a universal family of hash functions where $n - d \leq \ell < n$.[6]

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 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 $h(F_{\text{ltdf}}(s,x)) \in \{0,1\}^\ell$.

Lemma 3.5. *The algorithms (S_{crh}, F_{crh}) described above give a collection of collision-resistant hash functions from $\{0,1\}^n$ to $\{0,1\}^\ell$.*

[6] 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. For the purpose of contradiction, let $\mathcal{C}$ be an adversary that finds collisions for the collection with nonnegligible probability. Specifically, $\mathcal{C}$ takes an index $i = (s, h)$ and outputs some $x, x' \in \{0,1\}^n$. Consider the event that $x \neq x'$ form a valid collision in $F_{\text{crh}}(i, \cdot)$; because $F_{\text{ltdf}}(s, \cdot)$ is injective, this is equivalent to the event E that x, x' form a valid collision *and* $F_{\text{ltdf}}(s, x) \neq F_{\text{ltdf}}(s, x')$. (In the almost-always case, E also includes the constraint that $F_{\text{ltdf}}(s, \cdot)$ is actually injective, which fails to hold with only negligible probability.) Then it suffices to show that $p_0 = \Pr[E]$ when attacking the real construction 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 $h \leftarrow \mathcal{H}$. 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, via a statistical argument, that p_1 is negligible (even if $\mathcal{C}$ is *unbounded*).

Fix any 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, $|\mathcal{I}| \leq 2^{n-k}$. Now consider any fixed distinct pair $y, y' \in \mathcal{I}$: by universality of $\mathcal{H}$, we have $\Pr_h[h(y) = h(y')] = 2^{-\ell}$. Summing over all the (at most) $2^{2(n-k)} = 2^{n-2d}$ such pairs via a union bound, we see that

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

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 the proof is complete. □

Discussion. The crucial hypothesis in the above proof is that the residual leakage $n - k$ 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 at most $n - \omega(\log \lambda)$, because we only need to rule out collisions for the specific input selected by the adversary before the hash function is generated.

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 even 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 given above.

As described above, our constructions have the property that a trapdoor is known to (but supposedly discarded by) whomever generates the hash function, which may make it easy for that party to find collisions in the hash function. Formally, the construction should therefore be considered "private-coin," in contrast to a "public-coin" one for which it must remain hard to find a collision even given the random coins of the function generator. (See [40] for a detailed study of these two notions.) We point out that the alternate construction using S_{loss} also may not be public-coin, because knowing the random coins of S_{loss} may also make it easy to find collisions (and this is indeed the case for our concrete constructions in Sections 5 and 6).

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 security proof.

4.1 CPA-Secure Construction

Our CPA-secure cryptosystem may be seen as the result of applying the standard construction of an encryption scheme from a collection of injective trapdoor functions, using a universal hash family to instantiate the hard-core function that conceals the message. All the parameters in our CPA-secure system are functions of the security parameter λ; for notational convenience we usually omit this dependence.

Let $(S_{\text{ltdf}}, F_{\text{ltdf}}, F_{\text{ltdf}}^{-1})$ give a collection of (n, k)-lossy trapdoor functions (or almost-always lossy TDFs). Let $\mathcal{H}$ be a universal family of hash functions from $\{0,1\}^n$ to $\{0,1\}^\ell$, where $\ell \leq k - 2\lg(1/\epsilon)$ for some negligible $\epsilon = \mathsf{negl}(\lambda)$. The message space is $\{0,1\}^\ell$.

- **Key generation.** $\mathcal{G}$ generates an injective trapdoor function as $(s,t) \leftarrow S_{\text{inj}}$, and chooses a hash function $h \leftarrow \mathcal{H}$.

  The public key $pk = (s,h)$, and the secret key $sk = (t,h)$.

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

  It 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 a secret key $sk = (t,h)$ and a ciphetext $c = (c_1, c_2)$.

  It 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 universal 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 [25]. 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. Then the following is an d-out-of-t

(semi-honest) oblivious transfer protocol: the receiver generates d public keys normally (with decryption keys) and $t - d$ public keys obliviously, and delivers all t public keys to the sender, ordered so that the normal public keys correspond to the d desired messages. The sender encrypts each of the t messages under the corresponding public key and returns the t ciphertexts, and the receiver decrypts exactly the desired d.

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 note that the security properties in our protocol are dual to 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 only computational. We also point out that the EGL paradigm can also be used to construct semi-honest OT with existing lattice-based cryptosystems [2, 56, 57] for a similar reason: these cryptosystems are also 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.

Oblivious transfer protocol secure against *malicious* adversaries can be constructed using the zero-knowledge "compiler" paradigm of Goldreich, Micali, and Wigderson [34] or using a recent black-box transformation of Haitner [37], and secure multiparty computation can be obtained using the (non-black-box) compilation paradigm of Goldreich, Micali, and Wigderson [35]. However, these constructions are inefficient and primarily of theoretical interest. A recent work by Peikert, Vaikuntanathan, and Waters [52] 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.

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 $\{0,1\}^v$.[7] 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 branch set $B = \{0,1\}^v$, which contains the set of signature verification keys. (Almost-always lossy and ABO TDFs are also sufficient.)

We require that the total residual leakage of 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 universal family of hash functions from $\{0,1\}^n$ to $\{0,1\}^\ell$, where $0 < \ell \leq \kappa - 2\lg(1/\epsilon)$ for some negligible $\epsilon = \mathsf{negl}(\lambda)$. The message space is $\{0,1\}^\ell$.

- **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:

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

[7] As in prior schemes, we could also use a universal one-way hash function to hash the verification keys down to a smaller size; this would reduce the number of branches needed in the ABO. For simplicity we do not include this in our description of the system.

(In practice, the ABO trapdoor t' will never be used and may be discarded, but we retain it here for convenience in the security proof.)

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

  It generates one-time signature keypair $(vk, sk_\sigma) \leftarrow \mathsf{Gen}$, 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 output ciphertext is

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

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

  It 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.*

4.3.1 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 (arbitrary) fixed $b \in \{0,1\}$. 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 $b \in \{0,1\}$. Then by transitivity, the two chosen-ciphertext attack experiments for $b \in \{0,1\}$ are computationally indistinguishable, hence the cryptosystem is CCA2-secure.

We now define the sequence of games used 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. By default, each of these algorithms is unchanged from one game to the next, i.e., $\mathsf{Decrypt}_{i+1} = \mathsf{Decrypt}_i$, etc.

Before defining these algorithms for the individual experiments, we define one "global" aspect of the algorithms that applies to all the games: Setup always first chooses a one-time signature keypair $(vk^*, sk^*_\sigma) \leftarrow \mathsf{Gen}$, and then proceeds as described in the games below. Then 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 2.2.3, with the above-noted changes. That is, Setup_1 calls $(pk, sk) \leftarrow \mathcal{G}$ and outputs pk; $\mathsf{Decrypt}_1(c)$ outputs $\mathcal{D}(sk, c)$, and $\mathsf{Challenge}_1(m_0, m_1)$ outputs $c^* \leftarrow \mathcal{E}(pk, m_b)$.

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: The only change is in $\mathsf{Decrypt}_2$, which is defined as follows: given 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: 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: The only change is in $\mathsf{Decrypt}_4$, in which witness recovery 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 performs all the consistency checks of $\mathcal{D}$, and that the injective function trapdoor t is never used in this experiment.

The *full and final* description of $\mathsf{Decrypt}(c = (vk, c_1, c_2, c_3, \sigma))$ is now: if $vk = vk^*$, output $\perp$. Then if $\mathsf{Ver}(vk, (c_1, c_2, c_3), \sigma) \neq 1$, output $\perp$. Compute $x = G^{-1}_{\text{abo}}(t', vk, c_2)$, and check that $c_1 = F_{\text{ltdf}}(s, x)$ and $c_2 = G_{\text{abo}}(s', vk, x)$; if so, output $c_3 \oplus h(x)$ otherwise output $\perp$.

Game_5: 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 $(s, \perp) \leftarrow S_{\text{loss}}$.

The *full and final* description of Setup is now: choose $(vk^*, sk^*_\sigma) \leftarrow \mathsf{Gen}$, $(s', t') \leftarrow S_{\text{abo}}(vk^*)$, $(s, t) \leftarrow S_{\text{loss}}$, $h \leftarrow \mathcal{H}$, and output the public key $pk = (s, s', h)$.

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

The *full and final* description of $\mathsf{Challenge}(m_0, m_1)$ is now: ignoring the messages (and the bit b), let $x \leftarrow \{0,1\}^n$, $c_1 = F_{\text{ltdf}}(s, x)$, $c_2 = G_{\text{abo}}(s', vk^*, x)$, $c_3 \leftarrow \{0,1\}^\ell$, $\sigma \leftarrow \mathsf{Sign}(sk_\sigma, (c_1, c_2, c_3))$, and output $c = (vk^*, c_1, c_2, c_3, \sigma)$.

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, assuming the unforgeability of the signature scheme.

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 choosing $(pk, sk) \leftarrow \mathcal{G}$, 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 $\mathsf{Ver}(vk, (c_1, c_2, c_3), \sigma) = 1$ and (if after the challenge phase) $c \neq c^*$, $\mathcal{S}$ 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. (Note that the signing oracle is queried at most one time.)

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 $((c_1, c_2, c_3), \sigma) \neq ((c_1^*, c_2^*, c_3^*), \sigma^*)$. Therefore $\mathcal{S}$ outputs a valid 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 define a PPT simulator $\mathcal{S}$ that interacts in the hidden lossy branch experiment of the ABO collection. $\mathcal{S}$ generates $(vk^*, sk_\sigma^*) \leftarrow \mathsf{Gen}$, outputs (to the experiment) the two branches $(0^v, vk^*)$, and receives an ABO function index s' generated by either $S_{\text{abo}}(0^v)$ or $S_{\text{abo}}(vk^*)$. $\mathcal{S}$ then interacts with the CCA2 adversary $\mathcal{A}$, providing a view that is identical to either Game_2 or Game_3, respectively.

$\mathcal{S}$ implements Setup by executing 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 both Game_2 and Game_3 (each algorithm is unchanged between the two games). Note that $\mathcal{S}$ can do this because it knows the signing key sk^* the injective function trapdoor t.

By construction, 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 $S_{\text{abo}}(vk^*)$, and 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 $\mathsf{Decrypt}$. We show that $\mathsf{Decrypt}_3$ and $\mathsf{Decrypt}_4$ produce identical outputs (with overwhelming probability, if the lossy or ABO collections are almost-always).

First recall that if the lossy and ABO collections are almost-always, then the injectivity and invertibility 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 $vk \neq vk^*$). Thus there is a unique x such that $(c_1, c_2) = (f_s(x), g_{s',vk}(x))$. $\mathsf{Decrypt}_3$ finds that x by computing $F^{-1}_{\text{ltdf}}(t, c_1)$, while $\mathsf{Decrypt}_4$ 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 a function index 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 completes the implementation of Setup as in Game$_4$, by choosing $(vk^*, sk^*_\sigma) \leftarrow \mathsf{Gen}$, $(s', t') \leftarrow S_{\text{abo}}(vk^*)$, and $h \leftarrow \mathcal{H}$, and outputting a public key $pk = (s, s', h)$. (Note 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).

$\mathcal{S}$ implements Decrypt and Challenge just as in Game$_4$ and Game$_5$ (the algorithms are unchanged between the two games). Note that $\mathcal{S}$ can do this because it knows t', which is the only secret information $\mathsf{Decrypt}_4$ needs. 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 randomness (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 over the choice of h and x alone, the views in Game$_5$ and Game$_6$ are statistically indistinguishable; the claim follows by averaging over the rest of the randomness in the experiment.

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, we have

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

Now by the hypothesis that $\ell \leq \kappa - 2\lg(1/\epsilon)$ and Lemma 2.3, 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. □

4.3.2 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 an encryption witness.

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 [23], and is also technically similar to its use in the work of Canetti, Halevi, and Katz [17] in their construction of CCA2-secure encryption from identity-based cryptosystems. As discussion in the introduction, 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 choice of branch (from a branch set of super-logarithmic size). 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 straightforward changes to the proof. (In this case the description of h must also be signed under vk.) The same also 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

In this section we present constructions of lossy and all-but-one TDFs using groups in which the decisional Diffie-Hellman (DDH) problem is conjectured to be hard. The construction is a direct instantiation of the approach laid out informally in Section 1.3, and serves as a warm-up for the lattice-based constructions in the next section.

We proceed by recalling the DDH assumption and some basic operations on cyclic groups. Then we develop a special "matrix encryption" mechanism, and use it to construct lossy and all-but-one TDFs.

5.1 Background and Notation

Let $\mathcal{G}$ be a an algorithm that takes as input a security parameter λ and outputs a tuple $\mathbb{G} = (p, \langle G \rangle, g)$ where p is a prime, $\langle G \rangle$ is a description of a cyclic multiplicative group G of order p (i.e., an efficient algorithm for performing the group operation in G), 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 problem for $\mathcal{G}$ is to distinguish (with non-negligible advantage) between the ensembles

$$\left\{(\mathbb{G}, g^a, g^b, g^{ab})\right\}_{\lambda \in \mathbb{N}} \quad \text{and} \quad \left\{(\mathbb{G}, g^a, g^b, g^c)\right\}_{\lambda \in \mathbb{N}},$$

where $\mathbb{G} = (p, \langle G \rangle, g) \leftarrow \mathcal{G}(\lambda)$, and $a, b, c \leftarrow \mathbb{Z}_p$ are uniform and independent; the DDH assumption for $\mathcal{G}$ is that these ensembles are computationally indistinguishable. Particular families of groups in which the DDH assumption is conjectured to hold include the subgroup of quadratic residues in the group $\mathbb{Z}_q^*$ where $q = 2p + 1$ for prime p, and certain groups defined over elliptic curves.

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

We now define some notation that will be useful for our constructions and analysis. For a matrix $\mathbf{Y} = (y_{i,j}) \in \mathbb{Z}_p^{h \times w}$, we let $g^{\mathbf{Y}} = (g^{y_{i,j}}) \in G^{h \times w}$ be the matrix obtained by entry-wise exponentiation with base g. Then we can efficiently perform certain linear (homomorphic) operations on $\mathbf{Y}$ 'in the exponent,' namely:

1. Given $g^{\mathbf{Y}}$ and any matrix $g^{\mathbf{X}} \in G^{h \times w}$, we can compute

$$g^{\mathbf{X}+\mathbf{Y}} = (g^{x_{i,j}+y_{i,j}}) = (g^{x_{i,j}} \cdot g^{y_{i,j}}) \in G^{h \times w}$$

   by component-wise multiplication of $g^{\mathbf{X}}$ and $g^{\mathbf{Y}}$.

2. Given $g^{\mathbf{Y}}$ and any matrix $\mathbf{X} \in \mathbb{Z}_p^{h' \times h}$, let $\mathbf{Z} = \mathbf{XY} \in \mathbb{Z}_p^{h' \times w}$, where $z_{i,j} = \sum_{k \in [h]} x_{i,k} \cdot y_{k,j}$; we can compute $g^{\mathbf{Z}}$ by its entries

$$g^{z_{i,j}} = \left(\prod_{k \in [h]} (g^{y_{k,j}})^{x_{i,k}} \right) \in G.$$

   As special cases, we can left-multiply by a row vector $\mathbf{x} \in \mathbb{Z}_p^h = \mathbb{Z}_p^{1 \times h}$, or by any scalar $c \in \mathbb{Z}_p$.

5.2 Matrix Concealer

In this subsection we describe the foundation for our DDH-based lossy and ABO TDF constructions. This is a method for generating a pseudorandom "concealer" matrix (together with a trapdoor) that enjoys certain useful linearity properties. Namely, all the rows of the matrix lie in a one-dimensional subspace, and the trapdoor is a description of the subspace.

The algorithm $\mathsf{GenConceal}(h, w)$ works as follows: given positive integer dimensions $h, w = \text{poly}(\lambda)$,

- Choose $\mathbf{r} = (r_1, \ldots, r_n) \leftarrow \mathbb{Z}_p^h$ and $\mathbf{s} = (s_1, \ldots, s_w, 1) \leftarrow \mathbb{Z}_p^w \times \{1\}$ uniformly at random.
- Let $\mathbf{V} = \mathbf{r} \otimes \mathbf{s} = \mathbf{r}^t \mathbf{s} \in \mathbb{Z}_p^{h \times (w+1)}$ be the outer product of $\mathbf{r}$ and $\mathbf{s}$.
- Output $\mathbf{C} = g^{\mathbf{V}} \in G^{h \times (w+1)}$ as the concealer matrix, and $\mathbf{s}$ as the trapdoor.

We point out that if one truncates $\mathbf{s}$ to $\mathbf{s}' = (s_1, \ldots, s_w) \in \mathbb{Z}_p^w$, then $g^{\mathbf{r} \otimes \mathbf{s}'} = (g^{r_i \cdot s_j}) \in G^{h \times w}$ is exactly the DDH-based construction of a *pseudorandom synthesizer* due to Naor and Reingold [45]. (A synthesizer is a function $f(r, s)$ whose outputs are jointly pseudorandom on all pairs of r and s ranging over a polynomial number of uniformly random values $r_1, \ldots, r_h$ and $s_1, \ldots, s_w$, respectively. Here and in [45], the function $f : \mathbb{Z}_p \times \mathbb{Z}_p \to G$ is defined as $f(r, s) = g^{r \cdot s}$.)

Lemma 5.1. *Let $h, w = \text{poly}(\lambda)$. Under the DDH assumption for $\mathcal{G}$, the concealer matrix $\mathbf{C} = g^{\mathbf{V}}$ output by $\mathsf{GenConceal}(h, w)$ is pseudorandom over $G^{h \times w}$, i.e., computationally indistinguishable from the uniform distribution.*

Proof. We give a reduction from solving the DDH problem to distinguishing between $\mathbf{C} = g^{\mathbf{V}}$ and uniform. For the sake of simplicity and self-containment, we give a somewhat loose reduction; a tight reduction is also possible using the random self-reducibility of the DDH problem; see [45].

We first show a simpler fact that under the DDH assumption, $(g^{\mathbf{s}}, \mathbf{y} = g^{r \cdot \mathbf{s}})$ is computationally indistinguishable from $(g^{\mathbf{s}}, \mathbf{y} = g^{\mathbf{t}})$, where $r \leftarrow \mathbb{Z}_p$, $\mathbf{s} \leftarrow \mathbb{Z}_p^w \times \{1\}$, and $\mathbf{t} \leftarrow \mathbb{Z}_p^{w+1}$ are uniformly random and independent. To do so, define hybrid distributions $H_0, \ldots, H_w$, where (informally) in H_j the first j entries of $\mathbf{y}$ (and the last) are from $g^{r \cdot \mathbf{s}}$, while the remainder are uniform. More precisely, each H_j is generated as follows: choose uniform and independent $r \leftarrow \mathbb{Z}_p$, $\mathbf{s} \leftarrow \mathbb{Z}_p^w \times \{1\}$, and generate $\mathbf{y} \in G^{w+1}$ as $y_{w+1} = g^r$, $y_k = g^{r \cdot s_k}$ for $k \in [j]$, and $y_k \leftarrow G$ is uniform otherwise; output $(g^{\mathbf{s}}, \mathbf{y})$. It can be seen that H_w is distributed as $(g^{\mathbf{s}}, g^{r \cdot \mathbf{s}})$, whereas H_0 is distributed as $(g^{\mathbf{s}}, g^{\mathbf{t}})$.

We now show that for each $j \in [w]$, H_j and H_{j-1} are computationally indistinguishable under the DDH assumption, which by transitivity proves the claim. To do so, we give a reduction from distinguishing $(\mathbb{G}, g^r, g^s, g^{rs})$ from $(\mathbb{G}, g^r, g^s, g^t)$ (where $r, s, t \leftarrow \mathbb{Z}_p$ are uniform and independent) to distinguishing H_j from H_{j-1}. On input $(\mathbb{G}, g^r, g^s, y \in G)$, construct and output $(\mathbf{x} \in G^{w+1}, \mathbf{y} \in G^{w+1})$ as follows: let $x_{w+1} = g$, $y_{w+1} = g^r$; for $k \in [j-1]$ choose $s_k \leftarrow \mathbb{Z}_p$ and let $x_k = g^{s_k}$, $y_k = (g^r)^{s_k}$; for $k = j+1, \ldots, w$ choose $s_k \leftarrow \mathbb{Z}_p$ and let $x_k = g^{s_k}$, $y_k \leftarrow G$ uniformly; finally, let $x_j = g^s$, $y_j = y$. It may be verified by inspection that the reduction is correct, i.e., it maps $(\mathbb{G}, g^r, g^s, g^{rs})$ to H_j and $(\mathbb{G}, g^r, g^s, g^t)$ to H_{j-1}.

We now prove the lemma. Define (new) hybrid distributions $H_0, \ldots, H_h$ over matrices $\mathbf{C} \in G^{h \times (w+1)}$. In distribution H_i, the first i rows of $\mathbf{C}$ are computed as in $\mathsf{GenConceal}$, while the remaining rows are uniformly random in G^{w+1} (and independent of all other rows). Clearly H_h coincides with the output distribution of $\mathsf{GenConceal}$, and H_0 is the uniform distribution.

We now show that for all $i \in [h]$, H_i and H_{i-1} are computationally indistinguishable under the DDH assumption, which by transitivity proves the lemma. To do so, we give a reduction from distinguishing $(g^{\mathbf{s}}, g^{r \cdot \mathbf{s}})$ from $(g^{\mathbf{s}}, g^{\mathbf{t}})$ (where r, $\mathbf{s}$, and $\mathbf{t}$ are as above) to distinguishing H_i from H_{i-1}. Given $(g^{\mathbf{s}}, \mathbf{y} \in G^{w+1})$, generate a matrix $\mathbf{C}$ as follows: for each $k \in [i-1]$, choose $r_k \leftarrow \mathbb{Z}_p$ uniformly and let the kth row of $\mathbf{C}$ be $\mathbf{c}_k = (g^{\mathbf{s}})^{r_k} = g^{r_k \cdot \mathbf{s}}$. Let the ith row of $\mathbf{C}$ be $\mathbf{c}_i = \mathbf{y}$, and let the remaining rows be uniformly random and independent of all other rows. Clearly if $\mathbf{y} = g^{r \cdot \mathbf{s}}$, then the reduction's output is distributed as H_i, whereas if $\mathbf{y}$ is uniformly random then the output is distributed as H_{i-1}, as the proof is complete. □

We remark that according to the above proof, $g^{\mathbf{V}}$ remains pseudorandom even if the adversary is also given $g^{\mathbf{s}}$; this fact will not be needed by our construction.

5.3 Lossy TDF

We now describe our lossy TDF construction.

- *Sampling an injective/lossy function.* The function generators (for both the injective and lossy cases) first generate $\mathbb{G} = (p, \langle G \rangle, g) \leftarrow \mathcal{G}$, then invoke $\mathsf{GenConceal}(n, n)$ to generate a concealer matrix $\mathbf{C} = g^{\mathbf{V}} \in G^{n \times (n+1)}$ and trapdoor $\mathbf{s} \in \mathbb{Z}_p^{n+1}$.

  The injective function generator S_{inj} outputs function index $g^{\mathbf{Y}} = g^{\mathbf{V} + \mathbf{I}'}$, where $\mathbf{I}' \in \mathbb{Z}_p^{n \times (n+1)}$ is the $n \times n$ identity matrix extended by a zero column, i.e., the ith row of $\mathbf{I}'$ is the standard basis vector $\mathbf{e}_i \in \mathbb{Z}_p^{n+1}$. The trapdoor for the function is $\mathbf{s}$.

  The lossy function generation algorithm S_{loss} outputs function index $\mathbf{C} = g^{\mathbf{V}}$. There is no trapdoor output.

- *Evaluation algorithm.* F_{ltdf} takes as input $(g^{\mathbf{Y}}, \mathbf{x})$, where $g^{\mathbf{Y}} \in G^{n\times(n+1)}$ is a function index and $\mathbf{x} \in \{0,1\}^n \subset \mathbb{Z}_p^n$ is an n-bit input interpreted as a vector. The output is $\mathbf{z} = g^{\mathbf{xY}} \in G^{n+1}$, computed as described in Section 5.1.

  Note that if the function index $g^{\mathbf{Y}}$ was generated by S_{inj} (i.e., $\mathbf{Y} = \mathbf{r}^t\mathbf{s} + \mathbf{I}'$), then $z_{n+1} = g^{r'}$ where $r' = \mathbf{xr}^t \in \mathbb{Z}_p$, and for each $j \in [n]$ we have $z_j = g^{r's_j + x_j}$.

  In contrast, if $g^{\mathbf{Y}}$ was generated by S_{loss} (i.e., $\mathbf{Y} = \mathbf{r}^t\mathbf{s}$), then $\mathbf{z} = g^{r'\mathbf{s}}$ where $r' = \mathbf{xr}^t \in \mathbb{Z}_p$.

- *Inversion algorithm.* F_{ltdf}^{-1} takes as input $(\mathbf{s}, \mathbf{z})$ where $\mathbf{s}$ is the trapdoor information and $\mathbf{z}$ is the function output. The output $\mathbf{x} \in \{0,1\}^n$ is computed as follows: for each $j \in [n]$, compute $a_j = z_j / z_{n+1}^{s_j}$, and let $x_j \in \{0,1\}$ be such that $a_j = g^{x_j}$.

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

Proof. For a function generated by S_{inj} (the injective case), correctness of the inversion algorithm is apparent from the above remarks accompanying the evaluation and inversion algorithms.

For a function generated by S_{loss} (the lossy case), by the above remarks every output $\mathbf{z}$ is of the form $g^{r'\mathbf{s}}$ where $r' \in \mathbb{Z}_p$. Because $\mathbf{s}$ is fixed by the function index, there are at most $p = 2^{n-(n-\lg p)}$ distinct outputs of any particular lossy function, as desired.

Finally, indistinguishability of injective and lossy functions follows directly from Lemma 5.1: a lossy function index $g^{\mathbf{V}}$ is computationally indistinguishable from a uniformly random matrix over $G^{n\times(n+1)}$. The same is true for an injective function index $g^{\mathbf{V}+\mathbf{I}'}$ via the elementary reduction that adds $\mathbf{I}'$ in the exponent to its input matrix. This concludes the proof. □

Remarks.

1. Note that computing $F_{\text{ltdf}}(g^{\mathbf{Y}}, \mathbf{x} = \mathbf{0})$ always outputs $\mathbf{z} = g^{\mathbf{0}}$. This presents no problem, because it does not distinguish injective and lossy functions; the property holds for *any* function index $g^{\mathbf{Y}}$.

2. Our basic construction takes an n-bit input as a binary string and has an output of $n+1$ group elements, using an $n \times (n+1)$ matrix of group elements as the function index. It is possible to achieve somewhat smaller indices and outputs by using an $n' \times (n'+1)$ matrix and treating the input as an n'-dimensional vector of elements from a subset of $\mathbb{Z}_p$ of size 2^α, where $n' = \lceil n/\alpha \rceil$. However, this increases the running time of the inversion algorithm by about a 2^α factor, due to the enumeration over all possible values for each input block. Therefore, this generalization remains polynomial-time only for small values of α, i.e., $\alpha = O(\log \lambda)$.

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, and increases the leakage somewhat. Here we demonstrate a more efficient ABO collection where the number of branches can be as large as p. (It is also straightforward to combine this ABO construction with the generic one to get more than p branches.) The construction is a simple generalization of our lossy TDF construction.

Let the set of branches $B = \mathbb{Z}_p$.[8]

- *Sampling an ABO function.* The function generator $S_{\text{abo}}(b^* \in \mathbb{Z}_p)$ first generates $\mathbb{G} = (p, \langle G \rangle, g) \leftarrow \mathcal{G}$, then invokes $\mathsf{GenConceal}(n, n)$ to generate a concealer matrix $\mathbf{C} = g^{\mathbf{V}} \in G^{n \times (n+1)}$ and trapdoor $\mathbf{s}$.

  The function index is $g^{\mathbf{Y}} = g^{\mathbf{V} - b^* \mathbf{I}'}$, where $\mathbf{I}' \in \mathbb{Z}_p^{n \times (n+1)}$ is exactly as above in Section 5.3. The trapdoor information for the function is $t = (\mathbf{s}, b^*)$.

- *Evaluation algorithm.* G_{abo} takes as input $(g^{\mathbf{Y}}, b, \mathbf{x})$ where $g^{\mathbf{Y}}$ is a function index, $b \in \mathbb{Z}_p$ is the desired branch, and $\mathbf{x} \in \{0,1\}^n \subset \mathbb{Z}_p^n$ is an n-bit input interpreted as a vector. The output is $\mathbf{z} = g^{\mathbf{x}(\mathbf{Y} + b\mathbf{I}')} \in G^{n+1}$, computed using the homomorphisms as described in Section 5.1.

  Note that for $\mathbf{Y} = \mathbf{V} - b^* \mathbf{I}' = \mathbf{r}^t \mathbf{s} - b^* \mathbf{I}'$, we have $\mathbf{z} = g^{\mathbf{x}(\mathbf{V} + (b - b^*)\mathbf{I}')}$. In particular, $z_{n+1} = g^{r'}$ where $r' = \mathbf{x}\mathbf{r}^t \in \mathbb{Z}_p$, and for each $j \in [n]$ we have $z_j = g^{r' s_j + (b - b^*) x_j}$.

- *Inversion algorithm.* G_{abo}^{-1} takes as input $(t, b, \mathbf{y})$ where $t = (\mathbf{s}, b^*)$ is the trapdoor information, $b \neq b^*$ is the evaluated branch, and $\mathbf{z}$ is the function output. The output $\mathbf{x} \in \{0,1\}^n$ is computed as follows: for each $j \in [n]$, compute $a_j = z_j / z_{n+1}^{s_j}$, and let $x_j \in \{0,1\}$ be such that $a_j = g^{(b - b^*) x_j}$.

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

Proof. Correct inversion for $b \neq b^*$ and lossiness for $b = b^*$ follow directly from the above remarks and the same arguments as in the proof of Theorem 5.2.

The hidden lossy branch property (under the DDH assumption) also follows directly from Lemma 5.1: by an elementary reduction, for any branch $b^* \in \mathbb{Z}_p$ the first output of $S_{\text{abo}}(b^*)$ is computationally indistinguishable from uniform over $\mathbb{Z}_q^{n \times (n+1)}$. □

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 [57]. The LWE problem is a generalization to larger moduli of the *learning parity with noise* problem (see, e.g., [9]). It can be viewed as an (average-case) "bounded-distance decoding" problem on a certain family of random lattices under a natural error distribution. Interestingly, Regev showed that LWE is indeed hard on the average if standard lattice problems (like approximating the length of the shortest nonzero vector) are hard in the *worst case* for *quantum* algorithms [57]. Very recently, Peikert [50] gave a similar result via a *classical* reduction. No efficient (or even subexponential-time) quantum algorithms are known for the lattice problems in question, despite significant research efforts. Our constructions are based entirely on the LWE problem, and treat the connections to worst-case lattice problems [57, 50] as "black boxes."

Our lossy TDF based on LWE uses the same basic ideas as our DDH-based construction of Section 5: using linear homomorphisms, the function computes an encrypted linear product $\mathbf{x}\mathbf{M}$, where $\mathbf{M} = \mathbf{0}$ in the lossy case. However, we must overcome additional technical challenges stemming mainly from the fact that LWE involves extra random error terms that increase the leakage of the functions. Addressing this requires

[8] Strictly speaking, this does not conform to the ABO definition because the branch set should depend only on the security parameter λ. This issue can be addressed by letting the set of branches $B_\lambda = [q]$, where q does not exceed the *smallest* value of p produced by $\mathcal{G}$, and by interpreting a branch value $b \in B_\lambda$ modulo p when using a function defined over a group of order p.

some additional construction techniques, and some careful trade-offs between the lossy and injective cases. (See Section 6.3 for the lossy TDF construction.)

By calibrating the parameters appropriately, we can obtain lossy TDFs that lose any desired *constant fraction* (e.g., 99%) of the input. By itself, this is not 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 logarithm of the number of branches. Fortunately, we can also construct ABO TDFs directly from the LWE assumption by generalizing the lossy TDF construction with some additional ideas (see Section 6.4 for details).

6.1 Background

We start by introducing the relevant notation and computational problems, for the most part following [57].

For $x \in \mathbb{R}$, $\lfloor x \rceil = \lfloor x + 1/2 \rfloor$ denotes the nearest integer to x (with ties broken upward). Define $\mathbb{T} = \mathbb{R}/\mathbb{Z}$, i.e., the additive group of reals $[0, 1)$ with modulo 1 addition. We define an absolute value $|\cdot|$ on $\mathbb{T}$ as $|x| = \min\{\bar{x}, 1 - \bar{x}\}$, where $\bar{x} \in [0, 1)$ is the unique real-valued representative of $x \in \mathbb{T}$.

Probability distributions. The *Gaussian distribution* D_s with mean 0 and parameter s (sometimes called the *width*) is the distribution on $\mathbb{R}$ having density function $\exp(-\pi x^2/s^2)/s$. It is a standard fact that the sum of two independent Gaussian variables with mean 0 and parameters s_1 and s_2 (respectively) is a Gaussian variable with mean 0 and parameter $\sqrt{s_1^2 + s_2^2}$. We also need a standard tail inequality: for any $t \geq 1$, a Gaussian variable with parameter s has magnitude less than $t \cdot s$, except with probability at most $\exp(-t^2)$. Finally, it is possible to sample efficiently from a Gaussian distribution to any desired level of accuracy.

For $\alpha \in \mathbb{R}^+$ we define Ψ_α to be the distribution on $\mathbb{T}$ obtained by drawing $x \in \mathbb{R}$ from D_α and outputting $x \bmod 1$. For any probability distribution ϕ on $\mathbb{T}$ and integer $q \in \mathbb{Z}^+$ (often implicit) we define its *discretization* $\bar{\phi}$ to be the discrete distribution over $\mathbb{Z}_q$ given by $\lfloor q \cdot \phi \rceil \bmod q$.

Learning with errors (LWE). Let $d \geq 1$ be an integer dimension and $q \geq 2$ be an integer modulus. For a vector $\mathbf{s} \in \mathbb{Z}_q^d$ and a probability distribution χ over $\mathbb{Z}_q$, define $A_{\mathbf{s},\chi}$ to be the distribution on $\mathbb{Z}_q^d \times \mathbb{Z}_q$ obtained by choosing $\mathbf{a} \leftarrow \mathbb{Z}_q^d$ uniformly at random, $e \leftarrow \chi$, and outputting $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e)$.

The *learning with errors* problem $\mathsf{LWE}_{q,\chi}$, in its search version, is defined as follows. Given access to arbitrarily many independent samples from the distribution $A_{\mathbf{s},\chi}$ for some arbitrary $\mathbf{s} \in \mathbb{Z}_q^d$, find $\mathbf{s}$. The (average-case) *decision* version is to distinguish, with at least $1/\mathrm{poly}(d)$ advantage, between the distribution $A_{\mathbf{s},\chi}$ (for uniformly random $\mathbf{s} \in \mathbb{Z}_q^d$) and the uniform distribution over $\mathbb{Z}_q^d \times \mathbb{Z}_q$, given arbitrarily many samples.

In this paper we restrict our attention to the case of *prime* $q = \mathrm{poly}(d)$, for which the (worst-case) search and (average-case) decision versions of $\mathsf{LWE}_{q,\chi}$ are *equivalent* (up to a polynomial factor in the number of samples consumed) [57]. The $\mathsf{LWE}_{q,\chi}$ assumption is that these problems are hard; as shorthand, we can say that $A_{\mathbf{s},\chi}$ is pseudorandom (for uniformly random $\mathbf{s} \leftarrow \mathbb{Z}_q^d$).

For reasons that will become apparent below, the complexity of the LWE problem is measured primarily 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 others) implicitly be functions of d.

Connection to lattices. A (full-rank) d-dimensional *lattice* $\Lambda \subset \mathbb{R}^d$ is a discrete additive subgroup of $\mathbb{R}^d$. Equivalently, it is the set of all integer linear combinations of some d linearly independent basis vectors

$\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_d\} \subset \mathbb{R}^d$:

$$\Lambda = \left\{ \sum\nolimits_{i \in [d]} c_i \mathbf{b}_i \ : \ c_i \in \mathbb{Z} \right\}.$$

In computational settings, it is standard to represent a lattice by a basis.

Regev showed that for certain normal error distributions, the LWE problem 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 ([57])**.** *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 following two lattice problems in the* worst-case *to solving $\mathsf{LWE}_{q,\bar{\Psi}_\alpha}$:*

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

Proposition 6.1 has since been strengthened by Peikert in two ways: first [49], it also applies 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. Second [50], for $\alpha q \geq \sqrt{d \log d}$ there is also a *classical* (non-quantum) reduction from a variant of the GapSVP problem to LWE (and from GapSVP itself to LWE with modulus $q = 2^d$, but such a large value is not useful for our purposes).

The SIVP and GapSVP problems appear to be quite hard in the worst case (even for quantum algorithms): to obtain a $\mathrm{poly}(d)$ approximation factor, known algorithms require time and space that are exponential in d [4]; known polynomial-time algorithms obtain approximation factors that are only slightly subexponential in d [43, 61].

We 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 worst-case hardness guarantee.

6.2 Matrix Concealer

Here (as in Section 5.2) we describe a method for generating a special kind of pseudorandom "concealer" matrix (with trapdoor) that is the foundation for our lossy and ABO TDFs. Similarly to our DDH-based construction, the trapdoor is a description of a low-dimensional subspace, but unlike it, the rows of the concealer matrix only lie *near* the subspace.

The algorithm $\mathsf{GenConceal}_\chi(h, w)$ works as follows: given positive integer dimensions $h, w = \mathrm{poly}(d)$,

- Choose $\mathbf{A} \leftarrow \mathbb{Z}_q^{h \times d}$ and $\mathbf{S} \leftarrow \mathbb{Z}_q^{w \times d}$ uniformly at random, and $\mathbf{E} \leftarrow \chi^{h \times w}$.
- Output
$$\mathbf{C} = (\mathbf{A}, \mathbf{B} = \mathbf{A}\mathbf{S}^t + \mathbf{E}) \in \mathbb{Z}_q^{h \times (d+w)}$$
as the concealer matrix, and $\mathbf{S}$ as the trapdoor.

Lemma 6.2. *Let $h, w = \mathrm{poly}(d)$. Under the $\mathsf{LWE}_{q,\chi}$ assumption, the concealer matrix $\mathbf{C} = (\mathbf{A}, \mathbf{B})$ output by $\mathsf{GenConceal}_\chi$ is pseudorandom over $\mathbb{Z}_q^{h \times (d+w)}$, i.e., computationally indistinguishable from the uniform distribution.*

Proof. The proof proceeds by a simple hybrid argument over the w columns of $\mathbf{B}$. We define a sequence of hybrid distributions $H_0, \ldots, H_w$ over $\mathbb{Z}_q^{h \times (w+d)}$. In experiment H_j, the first j columns of $\mathbf{B}$ are as produced by $\mathsf{GenConceal}_\chi(h, w)$, while the remainder are uniformly random and independent. Clearly H_w is the output distribution of $\mathsf{GenConceal}_\chi(h, w)$ and H_0 is the uniform distribution.

We now show that for each $j \in [w]$, H_j and H_{j-1} are computationally indistinguishable under the $\mathsf{LWE}_{q,\chi}$ assumption, which by transitivity proves the claim. To do so, we give a reduction from distinguishing $A_{\mathbf{s},\chi}$ (for uniformly random $\mathbf{s} \leftarrow \mathbb{Z}_q^d$) from the uniform distribution U over $\mathbb{Z}_q^d \times \mathbb{Z}_q$ to distinguishing H_j from H_{j-1}. The reduction works as follows: given access to an unknown distribution D over $\mathbb{Z}_q^d \times \mathbb{Z}_q$, draw h samples $(\mathbf{a}_i, b_i)$ from D; form the matrix $\mathbf{A} \in \mathbb{Z}_q^{h \times d}$ whose ith rows is $\mathbf{a}_i$ and the column vector $\mathbf{b} \in \mathbb{Z}_q^{h \times 1}$ whose ith entry is b_i.

Choose $\mathbf{S} \leftarrow \mathbb{Z}_q^{w \times d}$ uniformly at random and $\mathbf{E} \leftarrow \chi^{h \times w}$, and let $\mathbf{B}' = \mathbf{A}\mathbf{S}^t + \mathbf{E}$. Form the matrix $\mathbf{B}$ that agrees with $\mathbf{B}'$ on its first $j-1$ columns, whose jth column is $\mathbf{b}$, and whose remaining columns are uniformly random and independent. Output $\mathbf{C} = (\mathbf{A}, \mathbf{B})$.

It should be clear by inspection that if the reduction's unknown distribution D is $A_{\mathbf{s},\chi}$, then the reduction's output is distributed as H_j, whereas if D is uniformly random then the reduction's output is distributed as H_{j-1}. This completes the proof. □

We now show a technical lemma that is needed for both the correctness and lossiness properties of our constructions.

Lemma 6.3. *Let h, w, p be positive integers. Let $q \geq 4ph$, let $1/\alpha \geq 8p(h+g)$ for some $g > 0$, and let $\chi = \bar{\Psi}_\alpha$. Then except with probability at most $w \cdot 2^{-g}$ over the choice of $\mathbf{E} \leftarrow \chi^{h \times w}$, the following holds: for every $\mathbf{x} \in \{0,1\}^h$, each entry of $(\mathbf{x}\mathbf{E})/q \in \mathbb{T}^w$ has absolute value less than $\frac{1}{4p}$.*

Proof. By a union bound over all w columns of $\mathbf{E}$, it suffices to show that for each column $\mathbf{e}^t \in \mathbb{Z}_q^{h \times 1}$ of $\mathbf{E}$, we have $|\langle \mathbf{x}, \mathbf{e} \rangle / q| < \frac{1}{4p}$ for all $\mathbf{x} \in \{0,1\}^h$ simultaneously, except with probability at most 2^{-g} over the choice of $\mathbf{e}$.

Below we show that for any *fixed* $\mathbf{x} \in \{0,1\}^h$,

$$\Pr_{\mathbf{e} \leftarrow \chi^h}\left[|\langle \mathbf{x}, \mathbf{e} \rangle / q| \geq \tfrac{1}{4p}\right] \leq 2^{-(h+g)}. \tag{2}$$

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

We now prove (2). By definition, $e_i = \lfloor q y_i \rceil \bmod q$ where $y_i \leftarrow D_\alpha$ are independent for all $i \in [h]$. Then by the triangle inequality, the distance (modulo 1) between $\langle \mathbf{x}, \mathbf{e} \rangle / q \in \mathbb{T}$ and $\langle \mathbf{x}, \mathbf{y} \rangle \in \mathbb{T}$ is at most $h/(2q) \leq 1/8p$. Therefore, it suffices to show that $|\langle \mathbf{x}, \mathbf{y} \rangle| < 1/8p$ except with probability at most $2^{-(h+g)}$.

Because the y_i are independent and $\mathbf{x}$ is binary, $\langle \mathbf{x}, \mathbf{y} \rangle$ is distributed as $D_{\alpha'}$ for $\alpha' \leq \sqrt{h} \cdot \alpha \leq \sqrt{h+g} \cdot \alpha$. Then by hypothesis on α and the tail inequality on Gaussian variables,

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

6.3 Lossy TDF

6.3.1 Overview

Our LWE-based lossy TDF is built upon the same main ideas as the DDH-based construction of Section 5. In particular, a function index is a concealed matrix $\mathbf{M}$, and the function is evaluated on $\mathbf{x} \in \{0,1\}^n$ by homomorphically computing an encrypted linear product $\mathbf{x}\mathbf{M}$ (where $\mathbf{M} = \mathbf{0}$ in the lossy case).

However, the use of LWE introduces several additional technical challenges. The chief difficulty is that the function outputs now also include accumulated error terms, so there are many more outputs than there would otherwise be if the error terms were not present. (The errors, of course, are necessary to securely conceal the underlying matrix $\mathbf{M}$.) The main challenge, therefore, is to limit the number of possible accumulated error vectors, while simultaneously ensuring (in the injective case) that all n bits of the input $\mathbf{x}$ can be recovered from the function output.

To start, it is helpful to see why the straightforward approach taken in our DDH-based construction does not yield a lossy function here. Recall that a lossy function index is simply an (unmodified) concealer matrix, which in the case of LWE is $\mathbf{C} = (\mathbf{A}, \mathbf{B} = \mathbf{A}\mathbf{S}^t + \mathbf{E}) \in \mathbb{Z}_q^{n \times (d+n)}$. Then given the first component $\mathbf{x}\mathbf{A}$ of the function output (and even ignoring its q^d possible values), the second component $\mathbf{x}\mathbf{B} = (\mathbf{x}\mathbf{A})\mathbf{S}^t + \mathbf{x}\mathbf{E} \in \mathbb{Z}_q^n$ *uniquely determines* $\mathbf{x}$ whenever $\mathbf{E}$ is invertible modulo q, which occurs often. Note that we do not have the luxury of restricting the rows of $\mathbf{E}$ to a low-dimensional subspace, because our proof of pseudorandomness (Lemma 6.2) requires the columns of $\mathbf{E}$ to be independent.

Our solution instead conceals suitably chosen *nonsquare* matrices of dimension $n \times w$, where $n \gg d + w$, to limit the number of outputs in the lossy case while still ensuring invertibility in the injective case. For the lossy case, the number of outputs $\mathbf{x}\mathbf{C} = (\mathbf{x}\mathbf{A}, \mathbf{x}\mathbf{B}) \in \mathbb{Z}_q^{d+w}$ is bounded by q^d times the number of possible values for $\mathbf{x}\mathbf{E}$, which we can bound using Lemma 6.3. In principle, the input length n can then be made as large as needed to obtain the desired amount of lossiness. (However, correct inversion will impose additional constraints.)

Of course, in the injective case we cannot conceal the identity matrix $\mathbf{I}$ because we are no longer using square matrices. Instead, we conceal a (suitably encoded) special matrix $\mathbf{G} \in \mathbb{Z}^{n \times w}$, whose rows are increasing power-of-2 multiples of each standard basis vector $\mathbf{e}_i \in \mathbb{Z}^w$. The idea is that we can easily recover $\mathbf{x} \in \{0,1\}^n$ from $\mathbf{x}\mathbf{G}$ via its base-2 representation. Of course, $\mathbf{G}$ is actually hidden by a concealer matrix over $\mathbb{Z}_q$, and we need to be able to compute $\mathbf{x}\mathbf{G}$ homomorphically and then decrypt it correctly. Computing $\mathbf{x}\mathbf{G}$ involves at most n row additions, and the entries of $\mathbf{x}\mathbf{G}$ are bounded by $2^{(n/w)}$, so we need to choose the parameters of the scheme to support such bounded homomorphism and message sizes. In the end, with a careful balancing of the parameters we can support any constant lossiness rate with a $\mathrm{poly}(d)$ input length n and modulus q, and a $1/\mathrm{poly}(d)$ noise rate α. (See Theorem 6.4 and Section 6.5 for the precise instantiation).

6.3.2 Construction

We now describe the lossy TDF construction, which involves several parameters. The dimension d is the main security parameter. As discussed above, our lossy TDF needs to support homomorphic operations resulting in encrypted integers as large as some p. We consider the modulus $p \geq 2$ (for simplicity, a power of 2) and matrix width $w \geq 1$ to be free variables, which will largely determine the lossiness of the functions. The input length of the functions is $n = w \lg p$. Finally, the modulus q and parameter α of the error distribution $\chi = \bar{\Psi}_\alpha$ of the underlying LWE problem will be instantiated later to ensure correct inversion. Looking ahead, the quantity $1/\alpha$ (which determines the worst-case approximation factor for lattice problems) will grow with p, so our goal will be to make p as small as possible while still ensuring lossiness.

Let $\mathbf{I} \in \mathbb{Z}^{w \times w}$ be the $w \times w$ identity matrix over the integers, and define a special row vector

$$\mathbf{p} = (2^0, 2^1, \ldots, 2^{\lg(p)-1} = p/2) \in \mathbb{Z}^{\lg p}$$

consisting of increasing powers of 2. Define the matrix $\mathbf{G} = \mathbf{I} \otimes \mathbf{p}^t \in \mathbb{Z}^{n \times w}$, where $\otimes$ denotes the tensor (or Kronecker) product. Essentially, the tensor product replaces each entry $e_{i,j}$ of $\mathbf{I}$ with the column vector $e_{i,j} \cdot \mathbf{p}^t \in \mathbb{Z}^{\lg p \times 1}$. Thus, the $((i-1)(\lg p) + j)$th row of $\mathbf{G}$ is $2^{j-1} \cdot e_i$, for $i, j \in [w]$.

To encrypt and operate homomorphically on integers as large as p using the LWE problem, we need to encode elements of $\mathbb{Z}_p$ suitably as elements of $\mathbb{Z}_q$. Define the encoding function $c : \mathbb{Z}_p \to \mathbb{Z}_q$ as

$$c(m) = \left\lfloor q \cdot \tfrac{m}{p} \right\rceil \in \mathbb{Z}_q,$$

where $\frac{m}{p} \in \mathbb{T}$, and extend c coordinate-wise to matrices over $\mathbb{Z}_p$. We also define a decoding function $c^{-1} : \mathbb{Z}_q \to \mathbb{Z}_p$ as

$$c^{-1}(v) = \left\lfloor p \cdot \tfrac{v}{q} \right\rceil \in \mathbb{Z}_p,$$

where again $\frac{v}{q} \in \mathbb{T}$ and we extend c^{-1} coordinate-wise to matrices over $\mathbb{Z}_q$. (Formally, c^{-1} is *not* the inverse function of c, but this abuse of notation evokes the intended use of the functions. For example, it can be verified that $c^{-1}(c(m)) = m$ for appropriate choices of p, q.)

- *Sampling an injective/lossy function.* The function generators (for both the injective and lossy cases) first invoke $\mathsf{GenConceal}_\chi(n, w)$ to generate a concealer matrix $\mathbf{C} = (\mathbf{A}, \mathbf{B} = \mathbf{A}\mathbf{S}^t + \mathbf{E}) \in \mathbb{Z}_q^{n \times (d+w)}$ and trapdoor $\mathbf{S}$.

  The injective function generator S_{inj} outputs function index $\mathbf{Y} = (\mathbf{A}, \mathbf{B} + \mathbf{M}) \in \mathbb{Z}_q^{n \times (d+w)}$ and trapdoor $\mathbf{S}$, where $\mathbf{M} = c(\mathbf{G} \bmod p)$.

  The lossy function generation algorithm S_{loss} simply outputs function index $\mathbf{Y} = \mathbf{C}$. There is no trapdoor output.

- *Evaluation algorithm.* F_{ltdf} takes as input $(\mathbf{Y}, \mathbf{x})$ where $\mathbf{Y}$ is a function index and $\mathbf{x} \in \{0,1\}^n$ is an n-bit input interpreted as a vector. The output is $\mathbf{z} = \mathbf{x}\mathbf{Y} \in \mathbb{Z}_q^{d+w}$.

  Note that if the function index $\mathbf{Y}$ was generated by S_{inj}, i.e., $\mathbf{Y} = (\mathbf{A}, \mathbf{A}\mathbf{S}^t + \mathbf{E} + \mathbf{M})$, then

  $$\mathbf{z} = (\mathbf{x}\mathbf{A}, (\mathbf{x}\mathbf{A})\mathbf{S}^t + \mathbf{x}(\mathbf{E} + \mathbf{M})).$$

  In contrast, if $\mathbf{Y}$ was generated by S_{loss}, i.e., $\mathbf{Y} = (\mathbf{A}, \mathbf{A}\mathbf{S}^t + \mathbf{E})$, then $\mathbf{z} = (\mathbf{x}\mathbf{A}, (\mathbf{x}\mathbf{A})\mathbf{S}^t + \mathbf{x}\mathbf{E})$.

- *Inversion algorithm.* F_{ltdf}^{-1} takes as input $(\mathbf{S}, \mathbf{z})$, where $\mathbf{S}$ is the trapdoor and $\mathbf{z} = (\mathbf{z}_1, \mathbf{z}_2) \in \mathbb{Z}_q^d \times \mathbb{Z}_q^w$ is the function output. It computes $\mathbf{v} = \mathbf{z}_2 - \mathbf{z}_1\mathbf{S}^t$, and lets $\mathbf{m} = c^{-1}(\mathbf{v}) \in \mathbb{Z}_p^w$. Finally, the output $\mathbf{x} \in \{0,1\}^n$ is computed as the unique binary solution to $\mathbf{x}\mathbf{G} = \bar{\mathbf{m}}$, where $\bar{\mathbf{m}} \in \mathbb{Z}^w$ is the unique integral vector congruent to $\mathbf{m}$ modulo p having entries in $\{0, \ldots, p-1\}$. (The solution $\mathbf{x}$ may be found efficiently by computing the base-2 representation of $\bar{\mathbf{m}}$.)

Theorem 6.4. *Let $q \geq 4pn = 4wp \lg p$ and $\chi = \bar{\Psi}_\alpha$ where $1/\alpha \geq 16pn = 16wp \lg p$. Then the algorithms described above define a collection of almost-always (n, k)-lossy TDFs under the $\mathsf{LWE}_{q,\chi}$ assumption, where the residual leakage $r = n - k$ is*

$$r \leq n \cdot \left(\frac{d}{w} + \left(\frac{d}{w} + 1 \right) \log_p(q/p) \right).$$

Note that in order for the residual leakage rate to be less than 1, we need both $w > d$ and $q < p^2$.

Proof. First, it follows directly from Lemma 6.2 that lossy function indices are computationally indistinguishable from injective ones, because both types are indistinguishable from uniform.

Now we show that with overwhelming probability over the choice of $\mathbf{Y} = (\mathbf{A}, \mathbf{B} = \mathbf{A}\mathbf{S}^t + \mathbf{E} + \mathbf{M})$ by S_{inj}, the inversion algorithm is correct for all $\mathbf{z} = (\mathbf{z}_1, \mathbf{z}_2) = F_{\text{ltdf}}(\mathbf{Y}, \mathbf{x}) = \mathbf{x}\mathbf{Y}$ where $\mathbf{x} \in \{0,1\}^n$. By the remarks accompanying the evaluation and inversion algorithms, we have

$$\mathbf{v} = \mathbf{z}_2 - \mathbf{z}_1\mathbf{S}^t = \mathbf{x}(\mathbf{E} + \mathbf{M}) = \mathbf{x}\mathbf{E} + \mathbf{x}\left\lfloor q \cdot \tfrac{\mathbf{G}}{p} \right\rceil.$$

Define the closed interval $I = [-\frac{1}{2}, \frac{1}{2}] \subset \mathbb{R}$. We have

$$\begin{aligned}
c^{-1}(\mathbf{v}) &= \left\lfloor p \cdot \left(\mathbf{x}\mathbf{E} + \mathbf{x}\left\lfloor q \cdot \tfrac{\mathbf{G}}{p} \right\rceil\right)/q \right\rceil \\
&\in \left\lfloor p \cdot (\mathbf{x}\mathbf{E})/q + (p/q) \cdot \mathbf{x}\left(I^{n\times w} + q \cdot \tfrac{\mathbf{G}}{p}\right) \right\rceil \\
&\in \left\lfloor \tfrac{1}{2} \cdot I^w + (p/q) \cdot \mathbf{x}I^{n\times w} + \mathbf{x}\mathbf{G} \right\rceil \qquad (3) \\
&\in \left\lfloor \tfrac{3}{4} \cdot I^w + \mathbf{x}\mathbf{G} \right\rceil \qquad (4) \\
&= \mathbf{x}\mathbf{G} \bmod p,
\end{aligned}$$

where (3) holds for all $\mathbf{x}$ simultaneously except with probability at most $w \cdot 2^{-n}$ over the choice of $\mathbf{E}$ by Lemma 6.3, and (4) is by the triangle inequality (yielding $\mathbf{x}I^{n\times w} \in n \cdot I^w$) and by the hypothesis that $q \geq 4pn$. Then because all the entries of $\mathbf{x}\mathbf{G} \in \mathbb{Z}^w$ are in $\{0, \ldots, p-1\}$ by definition of $\mathbf{G}$, the inversion algorithm successfully recovers $\mathbf{x}\mathbf{G}$ and outputs the correct $\mathbf{x} \in \{0,1\}^n$.

We now analyze the lossy functions. Let $\mathbf{Y} = (\mathbf{A}, \mathbf{B} = \mathbf{A}\mathbf{S}^t + \mathbf{E})$ be a function index produced by S_{loss}. Then for any input $\mathbf{x} \in \{0,1\}^n$,

$$F_{\text{ltdf}}(\mathbf{Y}, \mathbf{x}) = (\mathbf{z}_1, \mathbf{z}_2) = (\mathbf{x}\mathbf{A}, (\mathbf{x}\mathbf{A})\mathbf{S}^t + \mathbf{x}\mathbf{E}) \in \mathbb{Z}_q^d \times \mathbb{Z}_q^w.$$

The number of possible values for $\mathbf{z}_1 \in \mathbb{Z}_q^d$ is at most q^d, and given $\mathbf{z}_1$, the number of possible values for $\mathbf{z}_2$ is exactly the number of possible values for $\mathbf{x}\mathbf{E}$ (recall that $\mathbf{S}$ is fixed by the function index). By Lemma 6.3, the latter quantity is at most $(1 + q/2p)^w \leq (q/p)^w$. The total number of outputs of the function is therefore at most $q^d \cdot (q/p)^w$. The base-2 logarithm of this quantity is a bound on the residual leakage $r = n - k$:

$$\begin{aligned}
r &\leq d \cdot \lg q + w \cdot \lg(q/p) \\
&= n \cdot \frac{d \lg q}{w \lg p} + \frac{n}{\lg p} \cdot \lg(q/p) \\
&= n \cdot \left(\frac{d}{w} + \left(\frac{d}{w} + 1\right) \log_p(q/p)\right). \quad \square
\end{aligned}$$

6.4 All-But-One TDF

Our all-but-one TDF construction relies on all the ideas from the prior constructions, plus one additional technique. As always, evaluating the ABO function on $\mathbf{x} \in \{0,1\}^n$ involves computing an encrypted product $\mathbf{x}\mathbf{M}$, where $\mathbf{M}$ depends on the branch of the function being evaluated (and $\mathbf{M} = \mathbf{0}$ on the lossy branch).

In our DDH-based ABO function, the matrix $\mathbf{M}$ is some scalar multiple $(b - b^*)\mathbf{I}$ of the identity matrix, where $b, b^* \in \mathbb{Z}_p$. Because the matrices $\mathbf{M}$ are over the very large cyclic group $\mathbb{Z}_p$ (where p is the order of the DDH-hard group), the construction naturally supports a very large (super-polynomial) number of branches.

In the LWE setting, our matrices $\mathbf{M}$ may also be viewed over a group $\mathbb{Z}_p$. For correctness, however, the inverse error parameter $1/\alpha$ of the relevant LWE problem needs to grow with p, and the strength of the underlying complexity assumption (specifically, the approximation factor for worst-case lattice problems) in turn grows with $1/\alpha$. Consequently, simply using scalar multiples of the identity matrix $\mathbf{I}$ to get a large number of branches would induce a strong assumption, which is undesirable.

Instead, we generalize the set of branches to be a certain large linear family of integral matrices, for which every nonzero matrix in the family has full rank (i.e., its rows are linearly independent). The construction of such a family involves a simple linear encoding trick (a variant of which was used in [18] for different purposes) that maps a vector $\mathbf{v} \in \mathbb{Z}^w$ to a matrix $\mathbf{V} \in \mathbb{Z}_q^{w \times w}$ such that $\mathbf{V} = \mathbf{0}$ when $\mathbf{v} = \mathbf{0}$, and $\mathbf{V}$ is full-rank whenever $\mathbf{v} \neq \mathbf{0}$.[9] The full-rank property allows us to (efficiently) recover $\mathbf{x}$ from $\mathbf{xV}$ and knowledge of $\mathbf{v}$. Just as in the lossy TDF, the ABO function index conceals an "expanded" version of $\mathbf{V}$, namely $\mathbf{V} \otimes \mathbf{p}^t \in \mathbb{Z}^{w \lg p \times w}$, to achieve lossiness by way of increasing the input length. A final small detail is that for binary $\mathbf{x}$, the entries of $\mathbf{xV}$ can be as large as $w(p-1)$, rather than just $p-1$ when $\mathbf{V} = \mathbf{I} \otimes \mathbf{p}^t$, so we must enlarge the message space of the scheme accordingly.

Construction. Let the parameters $p \geq 2$ and $w \geq 2$ be free, assuming for simplicity that both p and w are powers of 2, and let $n = w \lg p$ as before. Again, the parameters q and α will be instantiated later to ensure correctness. The branch set $B = B_d = \{0,1\}^w$.

For an integral vector $\mathbf{v} = (v_1, \ldots, v_w) \in \mathbb{Z}^w$, define the "anti-cyclic" shift operation

$$s(\mathbf{v}) = (-v_w, v_1, \ldots, v_{w-1}) \in \mathbb{Z}^w.$$

(Note that the element v_w is negated in the output.) Also define the (anti-cyclic) shift matrix $\mathbf{V} = S(\mathbf{v}) \in \mathbb{Z}^{w \times w}$ whose ith row $\mathbf{v}_i = s^{(i-1)}(\mathbf{v})$ is the vector $\mathbf{v}$ shifted $i-1$ times. Clearly s (and hence S) is a linear operation, i.e., $s(\mathbf{v} + \mathbf{v}') = s(\mathbf{v}) + s(\mathbf{v}')$ and $s(c \cdot \mathbf{v}) = c \cdot s(\mathbf{v})$, so in particular $s(\mathbf{0}) = \mathbf{0}$. It is also the case that for any nonzero $\mathbf{v} \in \mathbb{Z}^w$, the matrix $S(\mathbf{v})$ is full-rank. The proof is as follows: because w is a power of 2, the polynomial $f(x) = x^w + 1$ is irreducible over $\mathbb{Q}[x]$. (This may be seen by applying Eisenstein's criterion with prime $p = 2$ to the shifted polynomial $f(x+1)$, or by noting that $f(x)$ is the $2m$th cyclotomic polynomial.) Then the vectors $\mathbf{v} \in \mathbb{Z}^w$ are in bijective correspondence with the polynomials $v(x) = v_1 + v_2 x + \cdots + v_w x^{w-1} \in \mathbb{Z}[x]/f(x)$, and $s(\mathbf{v})$ corresponds to $x \cdot v(x) \bmod f(x)$. Because the rows of $S(\mathbf{v})$ correspond to $v(x), x \cdot v(x) \ldots, x^{w-1} \cdot v(x) \bmod f(x)$ and $f(x)$ is irreducible, the rows are linearly independent when $\mathbf{v} \neq \mathbf{0}$.

As in the lossy TDF construction from Section 6.3.2 above, let $\mathbf{p} = (1, 2, \ldots, p/2) \in \mathbb{Z}^{\lg p}$ have increasing power-of-2 entries. Similarly, let $c : \mathbb{Z}_{p'} \to \mathbb{Z}_q$ and $c^{-1} : \mathbb{Z}_q \to \mathbb{Z}_{p'}$ be (respectively) the encoding and decoding functions, where we now use the modulus $p' = 2pw$. For notational simplicity, let $r : \{0,1\}^w \to \mathbb{Z}_q^{n \times w}$ be the function mapping a branch value to its encoded matrix over $\mathbb{Z}_q$:

$$r(\mathbf{v}) = c(S(\mathbf{v}) \otimes \mathbf{p}^t \bmod p').$$

- *Sampling an ABO function.* The function generator S_{abo} takes as input the desired lossy branch $\mathbf{v}^* \in \{0,1\}^w$. It runs $\mathsf{GenConceal}_\chi(n, w)$ to generate a concealer $\mathbf{C} = (\mathbf{A}, \mathbf{B} = \mathbf{AS}^t + \mathbf{E}) \in \mathbb{Z}_q^{n \times (d+w)}$ and trapdoor $\mathbf{S}$.

  The function index is $\mathbf{Y} = (\mathbf{A}, \mathbf{B} - r(\mathbf{v}^*)) \in \mathbb{Z}_q^{n \times (d+w)}$. The trapdoor information is $t = (\mathbf{S}, \mathbf{v}^*)$.

[9] We thank Daniele Micciancio for pointing out this avenue, which simplifies and slightly tightens the ABO construction from prior versions of this work.

- *Evaluation algorithm.* G_{abo} takes as input $(\mathbf{Y}, \mathbf{v}, \mathbf{x})$ where $\mathbf{Y} = (\mathbf{A}, \mathbf{B}')$ is a function index, $\mathbf{v} \in \{0,1\}^w$ is the desired branch, and $\mathbf{x} \in \{0,1\}^n$ is an n-bit input interpreted as a vector. The output is $\mathbf{x}(\mathbf{A}, \mathbf{B}' + r(\mathbf{v})) \in \mathbb{Z}_q^{d+w}$.

  Note that if $\mathbf{Y}$ was generated to have lossy branch $\mathbf{v}^*$, then we have

  $$(\mathbf{A}, \mathbf{B}' + r(\mathbf{v})) = (\mathbf{A}, \mathbf{A}\mathbf{S}^t + \mathbf{E} + (r(\mathbf{v}) - r(\mathbf{v}^*))) \approx (\mathbf{A}, \mathbf{A}\mathbf{S}^t + \mathbf{E} + r(\mathbf{v} - \mathbf{v}^*)),$$

  with equality when $\mathbf{v} = \mathbf{v}^*$. (We may have only a close approximation otherwise, because the function $c : \mathbb{Z}_{p'} \to \mathbb{Z}_q$ is not quite linear due to rounding.)

- *Inversion algorithm.* G_{abo}^{-1} takes as input $(t = (\mathbf{S}, \mathbf{v}^*), \mathbf{v}, \mathbf{z})$, where t is the trapdoor information, $\mathbf{v} \neq \mathbf{v}^*$ is the evaluated branch, and $\mathbf{z} = (\mathbf{z}_1, \mathbf{z}_2) \in \mathbb{Z}_q^d \times \mathbb{Z}_q^w$ is the function output. It computes $\mathbf{v} = \mathbf{z}_2 - \mathbf{z}_1\mathbf{S}^t$ and lets $\mathbf{m} = c^{-1}(\mathbf{v}) \in \mathbb{Z}_{p'}^w$. The output $\mathbf{x} \in \{0,1\}^n$ is computed as the unique binary solution to $\mathbf{x} \cdot (S(\mathbf{v} - \mathbf{v}^*) \otimes \mathbf{p}^t) = \bar{\mathbf{m}}$, where $\bar{\mathbf{m}} \in \mathbb{Z}^w$ is the unique integral vector congruent to $\mathbf{m}$ modulo $p' = 2pw$ having entries in $\{-w(p-1), \ldots, 0, \ldots, w(p-1)\}$.

  (The solution $\mathbf{x}$ may be found efficiently by solving the full-rank system $\mathbf{w} \cdot S(\mathbf{v} - \mathbf{v}^*) = \bar{\mathbf{m}}$, whose solution will be some integral $\mathbf{w} \in \{0, \ldots, p-1\}^w$, and then computing $\mathbf{w}$ in base 2.)

Theorem 6.5. *Let $q \geq 8p'n = 16w^2 p \lg p$ and $\chi = \bar{\Psi}_\alpha$ where $1/\alpha \geq 16p'n = 32w^2 p \lg p$. Then the algorithms described above define a collection of almost-always (n,k)-ABO TDFs with branch set $\{0,1\}^w$ under the $\mathsf{LWE}_{q,\chi}$ assumption, where the residual leakage $r = n - k$ is*

$$r \leq n \cdot \left(\frac{d}{w} + \left(\frac{d}{w} + 1 \right) \log_{p'}(q/p') \right).$$

Proof. The hidden lossy branch property (under the $\mathsf{LWE}_{q,\chi}$ assumption) follows directly from Lemma 6.2: by an elementary reduction, for any branch $\mathbf{v}^*$ the first output of $S_{\text{abo}}(\mathbf{v}^*)$ is computationally indistinguishable from the uniform distribution over $\mathbb{Z}_q^{n \times (d+w)}$.

Proving correct inversion on branch $\mathbf{v} \neq \mathbf{v}^*$ is very similar to the correctness proof for Theorem 6.4. Let $\mathbf{Y} = (\mathbf{A}, \mathbf{B}' = \mathbf{A}\mathbf{S}^t + \mathbf{E} - r(\mathbf{v}^*))$ be a function index generated by $S_{\text{abo}}(\mathbf{v}^*)$, and let

$$\mathbf{z} = (\mathbf{z}_1, \mathbf{z}_2) = G_{\text{abo}}(\mathbf{Y}, \mathbf{v}, \mathbf{x}) = \mathbf{x}(\mathbf{A}, \mathbf{A}\mathbf{S}^t + \mathbf{E} + r(\mathbf{v}) - r(\mathbf{v}^*)).$$

Then we have

$$\mathbf{v} = \mathbf{z}_2 - \mathbf{z}_1\mathbf{S}^t = \mathbf{x}\mathbf{E} + \mathbf{x}(r(\mathbf{v}) - r(\mathbf{v}^*)) \in \mathbf{x}\mathbf{E} + \{-n, \ldots, n\}^w + \mathbf{x} \cdot r(\mathbf{v} - \mathbf{v}^*) \in \mathbb{Z}_q^w,$$

where we have used the fact that $r(\mathbf{v}) - r(\mathbf{v}^*) \in r(\mathbf{v} - \mathbf{v}^*) + \{0, \pm 1\}^{n \times w}$ and the triangle inequality. Following the analysis from the proof of Theorem 6.4 (with the additional error terms of magnitude at most $n \leq \frac{q}{8p'}$), it can be shown that

$$c^{-1}(\mathbf{v}) = \mathbf{x} \cdot r(\mathbf{v} - \mathbf{v}^*) \bmod p'$$

except with negligible probability over the choice of $\mathbf{E}$. Because $\mathbf{v} - \mathbf{v}^*$ has entries from $\{0, \pm 1\}$, the integer entries of $\mathbf{x} \cdot r(\mathbf{v} - \mathbf{v}^*)$ lie in $\{-w(p-1), \ldots, w(p-1)\}$, so the inversion algorithm successfully recovers $\mathbf{x} \cdot r(\mathbf{v} - \mathbf{v}^*) \in \mathbb{Z}^w$ and outputs the correct $\mathbf{x}$.

Finally, the analysis of the residual leakage on the lossy branch is essentially identical to that in the proof of Theorem 6.4, with p replaced by p'. □

6.5 Instantiation and Worst-Case Connection

We now instantiate our schemes' parameters, and relate their security to worst-case lattice problems. The main outcome of this section is a connection between any desired lossiness rate $K \in (0,1)$ (larger K means more information is lost) and the associated approximation factor for lattice problems. This involves somewhat tedious (but otherwise routine) calculations to ensure that the parameters satisfy the various hypotheses of our theorems and Proposition 6.1. The dominating factor in the inverse error parameter $1/\alpha$ is $p \lg p$, so our main goal is to make p as small as possible subject to all the constraints.

Theorem 6.6. *Let $K \in (0,1)$ be any fixed constant. For any constant $c > \frac{3}{2(1-K)}$, there exist $w = \Theta(d)$, $p = O(d^c)$, prime $q = \tilde{O}(d^{c+3/2})$, and $1/\alpha = \tilde{O}(d^{c+1})$ with $\alpha q \geq \sqrt{d \log d}$ such that the construction of Section 6.3 yields a family of almost-always (n, Kn)-lossy TDFs (for all sufficiently large values of the security parameter d) under the $\mathsf{LWE}_{q,\bar{\Psi}_\alpha}$ assumption.*

The same applies for the construction in Section 6.4 of almost-always (n, Kn)-all-but-one TDFs, with $p' = O(d^c)$ replacing the condition on p.

In particular, by Proposition 6.1, the constructions are secure assuming that either SIVP or GapSVP are hard for quantum algorithms to approximate to within $\tilde{O}(d^{2+c})$ factors.

Proof. We select parameters that simultaneously satisfy all the constraints of Theorem 6.4 and Proposition 6.1 (and its subsequent variants). First, let

$$1/\alpha = 16pn = 16wp \lg p,$$

for w and p to be determined. For the worst-case connection to lattices, we let q be a prime in the range

$$q \in [1,2] \cdot \sqrt{d \log d}/\alpha = \tilde{\Theta}(d^{3/2}) \cdot p \lg p.$$

Next, we can let $w = \Theta(d)$ such that $\frac{d}{w} > 0$ is arbitrarily small. Then to obtain a residual leakage rate at most $R = 1 - K$, it suffices to choose p such that

$$\log_p(q/p) \leq R - \delta \iff (q/p) = \tilde{\Theta}(d^{3/2}) \cdot \lg p \leq p^{R-\delta}$$

for some constant $\delta > 0$. For any constant $c > \frac{3}{2R}$, there exists $p = O(d^c)$ for which the above is satisfied.

The analysis for the all-but-one TDF construction is identical, with p' replacing p. □

Acknowledgments

We are grateful to Dan Boneh for offering important insights in the early stages of this work, to Cynthia Dwork and Salil Vadhan for helpful comments, to Daniele Micciancio for suggesting a simpler construction of the LWE-based ABO, and to the anonymous STOC'08 and SICOMP 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 and Cynthia Dwork. The first and fourth public-key cryptosystems with worst-case/average-case equivalence. *Electronic Colloquium on Computational Complexity (ECCC)*, 14(97), 2007.

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

[5] 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.

[6] 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.

[7] Mihir Bellare, Dennis Hofheinz, and Scott Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In *EUROCRYPT*, pages 1–35, 2009.

[8] 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.

[9] 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.

[10] 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.

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

[12] 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.

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

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

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

[16] 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.

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

[18] Ronald Cramer and Ivan Damgård. On the amortized complexity of zero-knowledge protocols. In *CRYPTO*, pages 177–191, 2009.

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

[20] 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.

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

[22] 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.

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

[24] 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/`.

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

[26] 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.

[27] David Mandell Freeman, Oded Goldreich, Eike Kiltz, Alon Rosen, , and Gil Segev. More constructions of lossy and correlation-secure trapdoor functions. In *PKC*, 2010.

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

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

[30] 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.

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

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

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

[34] 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.

[35] 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.

[36] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

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

[38] 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.

[39] Dennis Hofheinz and Eike Kiltz. Secure hybrid encryption from weakened key encapsulation. In *CRYPTO*, pages 553–571, 2007.

[40] Chun-Yuan Hsiao and Leonid Reyzin. Finding collisions on a public road, or do secure hash functions need secret coins? In *CRYPTO*, pages 92–105, 2004.

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

[42] Gillat Kol and Moni Naor. Cryptography and game theory: Designing protocols for exchanging information. In *TCC*, pages 320–339, 2008.

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

[44] Petros Mol and Scott Yilek. Chosen-ciphertext security from slightly lossy trapdoor functions. In *PKC*, 2010.

[45] 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.

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

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

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

[49] 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.

[50] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In *STOC*, pages 333–342, 2009.

[51] Chris Peikert and Vinod Vaikuntanathan. Noninteractive statistical zero-knowledge proofs for lattice problems. In *CRYPTO*, pages 536–553, 2008.

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

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

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

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

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

[57] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), 2009. Preliminary version in STOC 2005.

[58] 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.

[59] Alon Rosen and Gil Segev. Chosen-ciphertext security via correlated products. In *TCC*, pages 419–436, 2009.

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

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

[62] Hovav Shacham. A Cramer-Shoup encryption scheme from the Linear Assumption and from progressively weaker Linear variants. Cryptology ePrint Archive, Report 2007/074, February 2007. `http://eprint.iacr.org/`.

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

[64] 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.

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