Lossy Trapdoor Functions and Their Applications

Chris Peikert and Brent Waters
presented by Di Yan

Shanghai Jiao Tong University

yandi821@sjtu.edu.cn

December 28, 2017
Overview

1. Preparation and Basic Concepts
   - Presentation Outline
   - Lossy Trapdoor Functions
   - All-But-One Trapdoor Functions
   - Basic Concepts

2. CCA-Secure Construction and Proof
   - From Lossy TDFs to ABO TDFs
   - CCA-Secure Construction
   - Proof of CCA Security

3. Realization from Lattices
   - Background Knowledge
   - Matrix Concealer
   - Construction of Lossy TDF
   - Review of the Presentation
Preparation and Basic Concepts
Introduce a new general primitive called lossy trapdoor functions
Construct All-But-One trapdoor functions from lossy TDFs
Achieve CCA-secure cryptosystems from lossy TDFs and ABO TDFs
Construct lossy trapdoor functions based on the hardness of the learning with errors (LWE) problems
In public-key cryptography in particular, two important notions are trapdoor functions (TDFs) and security under chosen ciphertext attack (CCA security).

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 and were first realized in the RSA function of Rivest, Shamir, and Adelman.

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.
λ is 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 \( r(\lambda) = n(\lambda) - k(\lambda) \).

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.

- **Easy to sample an injective function with trapdoor**: \( S_{\text{inj}} \) 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 \( f_s^{-1}(\cdot) \).
Easy to sample a lossy function: $S_{\text{loss}}$ outputs $(s, \bot)$ 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 $2^r = 2^{n-k}$.

Hard to distinguish injective from lossy: the first outputs of $S_{\text{inj}}$ and $S_{\text{loss}}$ are computationally indistinguishable. More formally, let $X_\lambda$ denote the distribution of $s$ from $S_{\text{inj}}$, and let $Y_\lambda$ denote the distribution of $s$ from $S_{\text{loss}}$. Then $X_\lambda \approx Y_\lambda$. 
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 $(S_{abo}, G_{abo}, G^{-1}_{abo})$ having the following properties:
Sampling a trapdoor function with given lossy branch: for any \( b^* \in B_\lambda \), \( S_{abo}(b^*) \) outputs \((s, t)\), where \( s \) is a function index and \( t \) is its trapdoor.

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

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

Hidden lossy branch: the views of any PPT adversary \( A \) in the following two experiments, indexed by a bit \( i \in \{0, 1\} \), are computationally indistinguishable: \( A \) outputs \((b_0^*, b_1^*) \in B_\lambda \times B_\lambda \) and is given a function index \( s \), where \((s, t) \leftarrow S_{abo}(b_i^*)\).
A signature scheme consists of three PPT algorithms $Gen$, $Sign$, and $Ver$, which are modeled as follows:

- $Gen$ outputs a verification key $v_k$ and a signing key $sk_\sigma$.
- $Sign(sk_\sigma, m)$ takes as input a signing key $sk_\sigma$ and a message $m \in \mathcal{M}$ and outputs a signature $\sigma$.
- $Ver(vk, m, \sigma)$ takes as input a verification key $vk$, a message $m \in \mathcal{M}$, and a signature $\sigma$, and outputs either 0 or 1.

**Definition**

A signature scheme $(\text{SigGen}, \text{Sign}, \text{Vrfy})$ is a one-time, strong signature scheme if for all ppt adversaries $\mathcal{A}$:

$$\Pr \left[ (vk, sk) \leftarrow \text{SigGen}(1^\kappa); m \leftarrow \mathcal{A}(vk); \sigma \leftarrow \text{Sign}(sk, m); (m', \sigma') \leftarrow \mathcal{A}(vk, \sigma); \text{Vrfy}(vk, (m', \sigma')) = 1 \land (m', \sigma') \neq (m, \sigma) \right] = \text{negl}(\kappa)$$
Randomness Extraction

**Lemma (Lemma 2.1)**

If $Y$ takes at most $2^r$ possible values and $X$ is any random variable, then 
$$\tilde{H}_\infty(X|Y) = H(X) - r$$

**Definition (Definition 2.2)**

A collection $\mathcal{H}$ of functions from $\{0, 1\}^n$ to $\{0, 1\}^l$ is an average-case $(n, k, l, \epsilon)$-strong extractor if for all pairs of random variables $(X, Y)$ such that $X \in \{0, 1\}^n$ and $\tilde{H}_\infty(X|Y) = k$, it holds that for $h \leftarrow \mathcal{H}$ and $r \leftarrow \{0, 1\}^l$, that 
$$\Delta(((h, h(X), Y), (h, r, Y))) \leq \epsilon$$

**Lemma (Lemma 2.3)**

Let $X, Y$ be random variables such that $X \in \{0, 1\}^n$ and $\tilde{H}_\infty(X|Y) = k$. Let $\mathcal{H}$ be a family of universal hash functions from $\{0, 1\}^n$ to $\{0, 1\}^l$, where $l \leq k - 2\lg(1/\epsilon)$. Then $\mathcal{H}$ is an average-case $(n, k, l, \epsilon)$-strong extractor.
Informally, a hard-core function for a function \( f : \{0, 1\}^n \rightarrow \{0, 1\}^* \) is a function \( h : \{0, 1\}^n \rightarrow \{0, 1\}^l \) such that \( h(x) \) is computationally indistinguishable from a uniformly random \( r \in \{0, 1\}^l \), given the value \( f(x) \).

In the following, let \((S_{\text{ltdf}}, F_{\text{ltdf}}, F_{\text{ltdf}}^{-1})\) give a collection of \((n, k)\)-lossy TDFs. Let \( \mathcal{H} \) be a universal family of hash functions from \( \{0, 1\}^n \) to \( \{0, 1\}^l \), where \( l \leq k - 2\lg(1/\epsilon) \) for some negligible \( \epsilon = \text{negl}(\lambda) \).

Define the following distributions that are generated by the experiments described below, which are implicitly indexed by the security parameter \( \lambda \).
Let $X_0, X_1, X_2, X_3$ be as defined above. Then

$$\{X_0\} \approx \{X_1\} \approx \{X_2\} \approx \{X_3\}$$

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

From Lemma 2.1 and Lemma 2.3
CCA-Secure Construction and Proof
Lemma (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 $(S_{abo}, G_{abo}, G_{abo}^{-1})$ give an $(n, k)$-ABO collection having branch set $\{0, 1\}$ (without loss of generality). We construct $(S_{inj}, S_{loss}, F_{ltdf}, F_{ltdf}^{-1})$ that give a collection of $(n, k)$-lossy TDFs as follows:

- The generator $S_{inj}$ outputs $(s, t) \leftarrow S_{abo}(1)$, and $S_{loss}$ outputs $(s, \perp)$ where $(s, t) \leftarrow S_{abo}(0)$.
- The evaluation algorithm $F_{ltdf}$ always evaluates on branch $b = 0$, i.e., $F_{ltdf}(s, x) = G_{abo}(s, 0, x)$.
- The inversion algorithm $F_{ltdf}^{-1}(t, y)$ outputs $x \leftarrow G_{abo}^{-1}(t, 0, y)$.

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

- The generator $S_{abo}(b^*)$ chooses $(s'_0, \perp) \leftarrow S_{loss}$, $(s'_1, t) \leftarrow S_{inj}$, and outputs $(s = (s'_{b^*}, s'_{1-b^*}), t)$.
- The evaluation algorithm $G_{abo}(s = (s_0, s_1), b, x)$ outputs $F_{ltdf}(s_b, x)$.
- The inversion algorithm $G_{abo}^{-1}(t, b, y)$ outputs $F_{ltdf}^{-1}(t, y)$. 
Lemma (Lemma 3.2)

If there exists an \((n, n - r)\)-ABO collection with branch set \(B = \{0, 1\}\), then for any \(l \geq 1\) there exists an \((n, n - l \cdot r)\)-ABO collection with branch set \(B = \{0, 1\}^l\).

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

- \(S_{abo}(b^*)\) generates \(\ell\) individual functions \((s_i, t_i) \leftarrow S(b^*_i)\) for \(i \in [\ell]\), where \(b^*_i\) is the \(i\)th bit of \(b^*\). The output is \((s = (s_1, \ldots, s_\ell), t = (b^*, t_1, \ldots, t_\ell))\).

- \(G_{abo}(s, b, x)\) computes \(y_i = G(s_i, b_i, x)\) for each \(i \in [\ell]\) and outputs \(y = (y_1, \ldots, y_\ell)\).

- \(G_{abo}^{-1}(t, b, y)\) 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 \(b = b^*\), \(G_{abo}^{-1}\) outputs \(\perp\).)
Let \((Gen, Sign, Ver)\) be a strongly unforgeable one-time signature scheme where the public verification keys are in \(\{0, 1\}^\nu\). Let \((S_{ltdf}, F_{ltdf}, F_{ltdf}^{-1})\) give a collection of \((n, k)\)-lossy trapdoor functions, and \((S_{abo}, G_{abo}, G_{abo}^{-1})\) give a collection of \((n, k')\)-ABO trapdoor functions having branch set \(B = \{0, 1\}^\nu\), 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

\[
 r + r' = (n - k) + (n - k') \leq n - \kappa
\]

for some \(\kappa = \kappa(n) = \omega(\log n)\). Let \(\mathcal{H}\) be a universal family of hash functions from \(\{0, 1\}^n\) to \(\{0, 1\}^l\), where \(0 < l \leq \kappa - 2\log(1/\epsilon)\) for some negligible \(\epsilon = negl(\lambda)\). The message space is \(\{0, 1\}^l\).
Key generation. $G$ 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:

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

(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 $(vk, sk_\sigma) \leftarrow \text{Gen}$, 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 $\sigma \leftarrow \text{Sign}(sk_\sigma, (c_1, c_2, c_3))$.

The output ciphertext is

$$c = (vk, c_1, c_2, c_3, \sigma).$$
Decryption. $D$ 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 $\text{Ver}(vk, (c_1, c_2, c_3), \sigma) = 1$; if not, it outputs ⊥. It then computes $x = F_{\text{ldf}}^{-1}(t, c_1)$, and checks that $c_1 = F_{\text{ldf}}(s, x)$ and $c_2 = G_{\text{abo}}(s', vk, x)$; if not, it outputs ⊥.

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

**Theorem (Theorem 4.2)**

*The algorithms $(G, E, D)$ described above give a CCA2-secure cryptosystem.*
Proof of CCA Security(1)

Claim 4.3

The adversary’s views in Game₁ and Game₂ are computationally indistinguishable, assuming the strong one-time existential unforgeability of the signature scheme.

Game₁: Algorithms Setup₁, Decrypt₁, and Challenge₁ are identical to those in the CCA2 experiment described in Section 2.2.3, with the above-noted changes. That is, Setup₁ calls \((pk, sk) \leftarrow \mathcal{G}\) and outputs \(pk\); Decrypt₁\((c)\) outputs \(D(sk, c)\), and Challenge₁\((m₀, m₁)\) outputs \(c^* \leftarrow \mathcal{E}(pk, m_b)\).

In particular, note that \(\mathcal{G}\) chooses the ABO lossy branch to be 0\(^v\), and \(D\) inverts \(c₁\) using the injective function trapdoor \(t\).

Game₂: The only change is in Decrypt₂, which is defined as follows: on input a ciphertext \(c = (vk, c₁, c₂, c₃, σ)\), if \(vk = vk^*\) (as chosen by Setup₂), then output \(⊥\). Otherwise return Decrypt₁\((c)\). (Note that by defining \(vk^*\) in Setup, this new rule is well-defined during both query phases.)
Proof of CCA Security (2)

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.

**Game$_2$:** The only change is in Decrypt$_2$, which is defined as follows: on input a ciphertext $c = (vk, c_1, c_2, c_3, \sigma)$, if $vk = vk^*$ (as chosen by Setup$_2$), then output ⊥. Otherwise return 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 $G$ we replace $(s', t') \leftarrow S_{abo}(0^v)$ with $(s', t') \leftarrow S_{abo}(vk^*)$.

Note that Decrypt$_3$ still decrypts using the injective function trapdoor $t$, and that the ABO function trapdoor $t'$ is never used in this experiment.
Proof of CCA Security(3)

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 $G$ we replace $(s', t') \leftarrow S_{abo}(0^v)$ with $(s', t') \leftarrow S_{abo}(vk^*)$.

Note that 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 Decrypt$_4$, in which witness recovery is now done using the ABO trapdoor $t'$. Formally, in $D$ we replace $x = F_{\text{ltdf}}^{-1}(t, c_1)$ with $x = G_{\text{abo}}^{-1}(t', vk, c_2)$. Note that Decrypt$_4$ still performs all the consistency checks of $D$, and that the injective function trapdoor $t$ is never used in this experiment.

The full and final description of Decrypt $(c = (vk, c_1, c_2, c_3, \sigma))$ is now: if $vk = vk^*$, output ⊥. Then if $\text{Ver}(vk, (c_1, c_2, c_3), \sigma) \neq 1$, output ⊥. Compute $x = G_{\text{abo}}^{-1}(t', vk, c_2)$, 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 ⊥.

**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).
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.
Claim 4.7

The adversary's views in Game\textsubscript{5} and Game\textsubscript{6} are statistically indistinguishable.
Realization from Lattices
Here we construct lossy TDFs based on the hardness of the learning with errors (LWE) problem. The chief difficulty is that the function outputs 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 $M$.) Thus the main challenge is to limit the number of possible accumulated error vectors, while simultaneously ensuring (in the injective case) that all $n$ bits of the input $x$ can be recovered from the function output.
For $x \in \mathbb{R}$, $\lfloor x \rfloor = \lfloor x + 1/2 \rfloor$ denotes the nearest integer to $x$. 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 $x \in \mathbb{T}$.

The Gaussian distribution $D_s$ with mean 0 and parameter $s$ (sometimes called the width) is the distribution on $\mathbb{R}$ having density function $\exp(-\pi x^2/s^2)/s$. 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 $\exp(-t^2)$.
We describe a method for generating a special kind of pseudorandom concealer matrix (with trapdoor) that is the foundation for our lossy TDFs.

The algorithm $\text{GenConceal}_\chi(n, w)$ works as follows: given positive integer dimensions $n, w = \text{poly}(d)$, $d$ is the main security parameter

- Choose $A \leftarrow \mathbb{Z}_q^{n \times d}$ and $S \leftarrow \mathbb{Z}_q^{w \times d}$ uniformly at random, and $E \leftarrow \chi^{n \times w}$.

- Output $C = (A, B = AS^t + E) \in \mathbb{Z}_q^{n \times (d+w)}$ as the concealer matrix, and $S$ as the trapdoor.
Lemma (Lemma 6.2)

Let $n, w = \text{poly}(d)$. Under the LWE_{q,\chi} assumption, the concealer matrix $C = (A, B)$ output by $\text{GenConceal}_\chi$ is pseudorandom over $\mathbb{Z}_q^{n \times (d + w)}$, i.e., computationally indistinguishable from the uniform distribution.

Lemma (Lemma 6.3)

Let $n, w, p$ be positive integers. Let $q \geq 4pn$, let $1/\alpha \geq 8p(n + g)$ for some $g > 0$, and let $\chi = \overline{\Phi}_\alpha$. Then except with probability at most $w \cdot 2^{-g}$ over the choice of $E \leftarrow \chi^{n \times w}$, the following holds: for every $x \in \{0, 1\}^n$, each entry of $(xE)/q \in \mathbb{T}^w$ has absolute value less than $1/4p$.
The input length of the functions is \( n = w \cdot \lg p \). Let \( I \in \mathbb{Z}^{w \times w} \) be the \( w \times w \) identity matrix over the integers, and define a special row vector

\[
P = (2^0, 2^1, \ldots, 2^{\lg p - 1} = p/2) \in \mathbb{Z}^{\lg p}
\]

consisting of increasing powers of 2. Define the matrix

\[
G = I \otimes p^t \in \mathbb{Z}^{n \times w}, \text{ where } \otimes \text{ denotes the tensor product.}
\]
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 $c : \mathbb{Z}_p \rightarrow \mathbb{Z}_q$ as

$$c(m) = \lfloor q \cdot \frac{m}{p} \rfloor \in \mathbb{Z}_q$$

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

$$c^{-1}(v) = \lfloor p \cdot \frac{v}{q} \rfloor \in \mathbb{Z}_p$$

where $\frac{v}{q} \in \mathbb{T}$ and we extend $c^{-1}$ coordinate-wise to matrices over $\mathbb{Z}_q$. 
**Sampling an injective/lossy function.** The function generators (for both the injective and lossy cases) first invoke $GenConceal_\chi(n, w)$ to generate a concealer matrix $C = (A, B = AS^t + E) \in \mathbb{Z}_q^{n \times (d+w)}$ and trapdoor $S$. The injective function generator $S_{inj}$ outputs function index $Y = (A, B + M) \in \mathbb{Z}_q^{n \times (d+w)}$ and trapdoor $S$, where $M = c(G \mod p)$. The lossy function generation algorithm $S_{loss}$ simply outputs function index $Y = C$. There is no trapdoor output.
**Evaluation algorithm.** $F_{ltdf}$ takes as input $(Y, x)$ where $Y$ is a function index and $x \in \{0, 1\}^n$ is an $n$-bit input interpreted as a vector. The output is $z = xY \in \mathbb{Z}_q^{d+w}$. Note that if the function index $Y$ was generated by $S_{inj}$, i.e., $Y = (A, AS^t + E + M)$, then

$$z = (xA, (xA)S^t + x(E + M)).$$

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

$$z = (xA, (xA)S^t + xE).$$
Construction of Lossy TDF(5)

- **Inversion algorithm.** $F_{ltdf}^{-1}$ takes as input $(S, z)$, where $S$ is the trapdoor and $z = (z_1, z_2) \in \mathbb{Z}_q^d \times \mathbb{Z}_q^w$ is the function output. It computes $v = z_2 - z_1 S^t$, and lets $m = c^{-1}(v) \in \mathbb{Z}_p^w$. Finally, the output $x \in \{0, 1\}^n$ is computed as the unique binary solution to $xG = \bar{m}$ (The solution $x$ may be found efficiently by computing the base-2 representation of $\bar{m}$.)

**Theorem (Theorem 6.4)**

Let $q \geq 4pn = 4wp \cdot lgp$ and $\chi = \bar{\Phi}_\alpha$ where $1/\alpha \geq 16pn = 16wp \cdot lgp$. Then the algorithms described above define a collection of almost-always $(n, k)$-lossy TDFs under the $LWE_{q,\chi}$ assumption, where the residual leakage $r = n - k$ is $r \leq n \cdot (\frac{d}{w} + (\frac{d}{w} + 1)\log_p(q/p))$. In order for the residual leakage rate to be less than 1, we need both $w > d$ and $q < p^2$. 
Construction of Lossy TDF(6)

Now we show that with overwhelming probability over the choice of \( \mathbf{Y} = (\mathbf{A}, \mathbf{B} = \mathbf{A}S^t + \mathbf{E} + \mathbf{M}) \) by \( S_{inj} \), the inversion algorithm is correct for all \( z = (z_1, z_2) = F_{ltdf}(\mathbf{Y}, x) = x\mathbf{Y} \) where \( x \in \{0, 1\}^n \). By the remarks accompanying the evaluation and inversion algorithms, we have

\[
\nu = z_2 - z_1 S^t = x(\mathbf{E} + \mathbf{M}) = x\mathbf{E} + x\lfloor q \cdot \frac{\mathbf{G}}{p} \rfloor
\]

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

\[
c^{-1}(\nu) = \lfloor p \cdot (x\mathbf{E} + x\lfloor q \cdot \frac{\mathbf{G}}{p} \rfloor)/q \rfloor
\]

\[
\in \lfloor p \cdot (x\mathbf{E})/q + (p/q)x(l^{n \times w} + q \cdot \frac{\mathbf{G}}{p}) \rfloor
\]

\[
\in \lfloor \frac{1}{2} \cdot l^w + (p/q)x l^{n \times w} + x\mathbf{G} \rfloor
\]

\[
\in \lfloor \frac{3}{4} \cdot l^w + x\mathbf{G} \rfloor
\]

\[
= x\mathbf{G} mod p
\]
Construction of Lossy TDF(7)

For any input \( x \in \{0, 1\}^n \),
\[
F_{\text{ltdf}}(Y, x) = (z_1, z_2) = (xA, (xA)S^t + xE) \in \mathbb{Z}_q^d \times \mathbb{Z}_q^d.
\]
The number of possible values for \( z_1 \in \mathbb{Z}_q^d \) is at most \( q^d \), and given \( z_1 \), the number of possible values for \( z_2 \) is exactly the number of possible values for \( xE \) (recall that \( S \) is fixed by the function index). 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 \( q^d \cdot (q/p)^w \). In order to make the input size larger than the output size to ensure lossiness, we have \( 2^n > q^d \cdot (q/p)^w \):

\[
n > d \cdot \lg q + w \cdot \lg q/p
= n \cdot \frac{d \cdot \lg q}{w \cdot \lg p} + \frac{n \cdot \lg q}{\lg p} \cdot \frac{1}{\lg p}
= n \cdot \left( \frac{d}{w} + \left( \frac{d}{w} + 1 \right) \log_p(q/p) \right)
\]
Introduce a new general primitive called lossy trapdoor functions
Construct All-But-One trapdoor functions from lossy TDFs
Achieve CCA-secure cryptosystems from lossy TDFs and ABO TDFs
Construct lossy trapdoor functions based on the hardness of the learning with errors (LWE) problems
References

Chris Peikert (2008)

Lossy Trapdoor Functions and Their Applications

The End