Cryptography Reading Group
Lossy Trapdoor Functions and Their Applications
By Chris Peikert, Brent Waters

Presented by Josh Hoak

January 25, 2010

1 Trapdoor Functions and Lossy Trapdoor functions

Trapdoor functions provide the basis for public key cryptography. Essentially, they are one-way functions that provide a *trapdoor* – an easy way to invert (if you know the trapdoor). In other words, we require for trapdoor functions that:

1. They are easy to compute in one direction
2. They are difficult to compute in the reverse direction without the trapdoor

Multiplication is the most common example of a function we believe to be one-way, in that it is much easier to multiply than it is to factor (or at least, so we believe). This gives rise to several common problems used in cryptography.

Example: The Discrete Log Problem
Given: g a generator for some cyclic group G and g^x, for some $x \in G$
Find: x

Example: Computational Diffie-Hellman Problem
Given: g a generator for some cyclic group G, g^x, and g^y, for $x, y \in G$
Find: g^{xy}

Example: Decisional Diffie-Hellman
Given: g a generator for some cyclic group G, g^x, g^y, g^z for $x, y, z \in G$
Find: Is $z \equiv xy \mod |G|$

Example: Factor Problem
Given: pq, for primes p, q
Find: Find p and q

1.1 Trapdoor Functions

Formally, we talk about collections of trapdoor functions. A collection of *trapdoor functions* is given by the triple of algorithms running in probabilistic-polynomial-time (PPT): (S, F, F^{-1}).

- $S \rightarrow (s, t)$. Algorithm S produces a function index s and the trapdoor t
- $F(s, \cdot) \rightarrow f_s(\cdot)$. Algorithm F takes s and a message input $(\cdot)$ and computes $f_s(\cdot)$, where $f_s : \{0,1\}^n \rightarrow \{0,1\}^n$. We require that for a given s, that F be injective.

– $F^{-1}(t, \cdot) \rightarrow f_s^{-1}(\cdot)$. Computes the inverse of F as you would expect. Were $f_s(\cdot)$ not injective, $F^{-1}(t, \cdot)$ would be impossible.

Note that for any PPT inverter $\mathcal{I}$, we require $\Pr(\mathcal{I}(s, f_s(x)) \rightarrow x)$ is negligible for a collection of functions to be trapdoor functions.

1.2 Lossy-Trapdoor Functions

Informally, we think of lossy-trapdoor functions as non-injective trapdoor functions (remember that we are thinking about collections of functions).

We specify a collection of Lossy-Trapdoor Functions by the 4-tuple:

$$(S_{\text{inj}}, S_{\text{loss}}, F_{\text{ltdf}}, F_{\text{ltdf}}^{-1},)$$

We define:

$$\begin{aligned} \lambda : &\quad \text{The security parameter} \\ n(\lambda) = \text{poly}(\lambda) : &\quad \text{the input length of the function} \\ k(\lambda) \leq n(\lambda) : &\quad \text{the lossiness} \\ r(\lambda) = n(\lambda - k(\lambda) : &\quad \text{the leakage} \end{aligned}$$

For a collection of functions to be deemed a collection of *Lossy-Trapdoor Functions*, three properties must hold:

1. Easy to sample injective functions with trapdoor: $S_{\text{inj}} \rightarrow (s, t)$ where s is a function index and t is its trapdoor. When S_{inj} outputs such a pair, F_{ltdf} and F_{ltdf}^{-1} work as in standard trapdoor examples.
2. Easy to sample a lossy function: $S_{\text{loss}} \rightarrow (s, \perp)$, where s is a function index, and F_{ltdf} computes $f_s(\cdot)$, where $f_s : \{0,1\}^n \rightarrow \{0,1\}^r$ recalling that $r = n - k$.
3. Hard to distinguish injective from lossy: we require that S_{inj} and S_{loss} be computationally indistinguishable. Formally, if X_λ denotes the distribution of s from S_{inj} and if Y_λ denotes the distribution of s from S_{loss}, then $\{X_\lambda\} \stackrel{c}{\approx} \{Y_\lambda\}$.

Important note! We do explicitly require that an injective function be hard to invert.

1.3 Computational Indistinguishability

In the definition for lossy-TDFs, computational indistinguishability (CI) plays a key role – (represented by $\{X_\lambda\} \stackrel{c}{\approx} \{Y_\lambda\}$). Here, as an aside, I present formally the requirements for CI.

Definition 1 (Statistical distance). *Let X and Y be random variables over a countable set S. Then, we define statistical distance (notated $\Delta(X, Y)$) as:*

$$\Delta(X, Y) := \frac{1}{2} \sum_{s \in S} \mid \Pr[X = s] - \Pr[Y = s] \mid$$

Definition 2 (Statistical Indistinguishability). *Let $\mathcal{X} = \{X_\lambda\}_{\lambda\in\mathbb{N}}, \mathcal{Y} = \{Y_\lambda\}_{\lambda\in\mathbb{N}}$ be two ensambles of random variables indexed by λ. Then $\mathcal{X}$ and $\mathcal{Y}$ are statistical indistinguishability (notated $\{X_\lambda\} \overset{s}{\approx} \{Y_\lambda\}$) when the statistical distance is negligible. In symbols:*

$$\{X_\lambda\} \overset{s}{\approx} \{Y_\lambda\} \quad \text{if} \quad \Delta(X, Y) = \mathsf{negl}(\lambda)$$

For the following, I assume familiarity with the *advantage* for an algorithm (adversary) $\mathcal{A}$.

Definition 3 (Computational Indistinguishability). *Let $\mathcal{X}, \mathcal{Y}$ be defined as above and also let there be some probabilistic polynomial time algorithm $\mathcal{A}$. Then, we say that $\mathcal{X}, \mathcal{Y}$ are computationally indistinguishable (notated $\{X_\lambda\} \overset{c}{\approx} \{Y_\lambda\}$) if the advantage of any PPT algorithm $\mathcal{A}$ is $\mathsf{negl}(\lambda)$.*

2 Lossy Trapdoor Functions are Trapdoor Functions

Lemma 1. *Let $(S_{\text{ltdf}}, F_{\text{ltdf}}, F^{-1}_{\text{ltdf}})$ give a collection of (n, k)-lossy trapdoor functions. Let $k \geq \omega(\log \lambda)$. Then, $(S_{\text{inj}}, F_{\text{ltdf}}, F^{-1}_{\text{ltdf}})$ give a collection of injective-trapdoor functions (in the conventional sense).*

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

The rest of the proof proceeds by way of contradiction. Suppose that $\mathcal{I}$ is PPT inverter for the collection of functions described above. Then, we can use $\mathcal{I}$ to build a distinguisher $\mathcal{D}$ that distinguishes injective functions and lossy ones. This is a contradiction, since we require that the lossy-functions and injective-functions be indistinguishable.

More formally: If $\mathcal{I}$ is an inverter, then $\mathcal{I}(s, f_s(x))$ outputs x with a non-negligible probability. From inverter $\mathcal{I}$ we construct $\mathcal{D}$ as follows:

Algorithm $\mathcal{D}$:
On input (i.e. a function index s), choose $x \to \{0, 1\}^n$.
Compute $y = F_{\text{ltdf}}(s, x)$
Let $x' \leftarrow \mathcal{I}(s, y)$
If $x' = x$, output 1(injective); otherwise, output 0 (lossy).

Now, we need to analyze $\mathcal{D}$.

(1) If s is generated by S_{inj}, $\mathcal{D}$ outputs 'injective' since, by assumption, $\mathcal{I}$ outputs x with non-negligible probability

(2) The slipperiness comes in the case in which we let s be any fixed function index generated by S_{loss}. The probability that even an unbounded $\mathcal{I}$ predicts x is given by the average *min-entropy* of x conditioned on $f_s(x)$. In other words, the predictability is given by at most $2^{-\tilde{H}_\infty(x|f_s(x))}$. Since $f_s(\cdot)$ takes at most 2^{n-k} values. The Entropy Approx. Lemma gives us:

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

Since $k = \omega(\log \lambda)$, 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}. ✦

2.1 Reference

Definition 4 (Min-entropy). *Let X be random variable over a domain S. We define the min-entropy as:*

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

Lemma 2 (Entropy Approximation Lemma). *If Y takes at most 2^r possible values and Z is any random variable, then*

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