More Constructions of Lossy and Correlation-Secure Trapdoor Functions

David Mandell Freeman[1], Oded Goldreich[2],
Eike Kiltz[3], Alon Rosen[4], and Gil Segev[2]

[1]Stanford University, USA
[2]Weizmann Institute of Science, Israel
[3]CWI, Netherlands
[4]IDC Herzliya, Israel

PKC 2010
Paris, France
27 May 2010

LTDFs: What are they?

Functions $f(x)$ that can behave in two ways [PW08]:

1. **Injective:**
   - $f(x)$ is 1-to-1.
   - There is a *trapdoor* that allows $f(x)$ to be inverted.
2. **Lossy:**
   - $f(x)$ loses information: image is smaller than domain.
   - If $|Domain| = 2^n$ and $|Image| = 2^{n-\ell}$,
     $f(x)$ has *ℓ bits of lossiness*.

Security: descriptions of injective functions and lossy functions are *computationally indistinguishable*.

LTDFs: What are they good for?

Modular constructions of cryptographic primitives:

- Collision-resistant hash functions [PW08]
- Oblivious transfer [PW08]
- CCA-secure public-key encryption [PW08]
- Deterministic public-key encryption [BFO08]
- Security against selective opening attacks [BHY09]
- and others...

Given all these uses, we'd like to have a big "library" of LTDFs based on different computational assumptions.

Constructions of LTDFs

[PW08] construct LTDFs based on:

1. *Decision Diffie-Hellman* assumption (DDH)
2. *Learning With Errors* assumption (LWE) on lattices

We add new constructions based on:

3. *Quadratic Residuosity* assumption (QR)
   - Apparently weaker than $2vs3primes$ of [MY10].
   - Generalized to eth power residuosity in full version.
4. *Composite Residuosity* assumption (Paillier)
   - Discovered concurrently and independently by [BFO08].
5. *d-Linear* assumption
   - Simplifies and generalizes DDH construction of [PW08].

Correlation-Secure Trapdoor Functions

Generalization of one-way function to correlated inputs [RS09]:

Given collection of functions $\mathcal{F}$ and distribution $\mathcal{C}$ on $Domain(\mathcal{F})^k$ correlation-security says that for

$$f_1, \ldots, f_k \stackrel{R}{\leftarrow} \mathcal{F} \quad \text{and} \quad (x_1, \ldots, x_k) \stackrel{R}{\leftarrow} \mathcal{C},$$

the function $(x_1, \ldots, x_k) \mapsto \big(f_1(x_1), \ldots, f_k(x_k)\big)$ is one-way.

- Can be used to construct CCA-secure public key encryption.
- Implied by LTDFs (with any amount of lossiness [MY10]).
- Our contribution: new construction based on the hardness of *syndrome decoding*.

Outline

1. Lossy and Correlation-Secure Trapdoor Functions
2. LTDFs from Quadratic Residuosity
3. LTDFs from d-Linear assumptions

Some (old) observations

$N = PQ$, $P \equiv Q \equiv 3 \bmod 4$ prime.

- Squaring function $x \mapsto x^2 \bmod N$ is lossy:
  - 4-to-1 map on $\mathbb{Z}_N^* \Rightarrow 2$ bits of lossiness.
- However, x^2 can be inverted if we know
  1. *Jacobi symbol* $\mathrm{JS}_N(x) \in \{-1, 0, 1\}$
  2. *Sign* of $x \bmod N$ (represented as integer in $-N/2, \ldots, N/2$)
- Specifically, if $(\pm x_0, \pm x_1)$ are 4 square roots of $y \bmod N$, then

$$\mathrm{JS}_N(x_0) = \mathrm{JS}_N(-x_0) = -\mathrm{JS}_N(x_1) = -\mathrm{JS}_N(-x1).$$

So to get injective function from squaring, encode 2 extra bits in the output (e.g., Williams).

Creating an injective function

Problem: how to encode extra bits in a computationally indistinguishable way.

Solution: put them in the *exponent* of *quadratic non-residues*.

Define:

$$h(x) := \begin{cases} 1, & \text{if } x \bmod N > 0 \\ 0, & \text{if } x \bmod N < 0 \end{cases} \qquad j(x) := \begin{cases} 1, & \text{if } \mathrm{JS}_N(x) = -1 \\ 0, & \text{if } \mathrm{JS}_N(x) = 0 \text{ or } 1 \end{cases}$$

Choose $r, s \in \mathbb{Z}_N^*$ with $\mathrm{JS}_N(r) = -1$, $\mathrm{JS}_N(s) = 1$, s a quadratic non-residue. Then injective function is

$$f_{r,s,N}(x) = x^2 \cdot r^{j(x)} \cdot s^{h(x)} \bmod N.$$

Creating an injective function

Problem: how to encode extra bits in a computationally indistinguishable way.

Solution: put them in the *exponent* of *quadratic non-residues*.

Define:

$$h(x) := \begin{cases} 1, & \text{if } x \bmod N > 0 \\ 0, & \text{if } x \bmod N < 0 \end{cases} \qquad j(x) := \begin{cases} 1, & \text{if } \mathrm{JS}_N(x) = -1 \\ 0, & \text{if } \mathrm{JS}_N(x) = 0 \text{ or } 1 \end{cases}$$

Choose $r, s \in \mathbb{Z}_N^*$ with $\mathrm{JS}_N(r) = -1$, $\mathrm{JS}_N(s) = 1$, s a quadratic non-residue. Then injective function is

$$f_{r,s,N}(x) = x^2 \cdot r^{j(x)} \cdot s^{h(x)} \bmod N.$$

Recovering the extra bits

$$f_{r,s,N}(x) = x^2 \cdot r^{j(x)} \cdot s^{h(x)} \bmod N$$

$$h(x) := \begin{cases} 1, & \text{if } x \bmod N > 0 \\ 0, & \text{if } x \bmod N < 0 \end{cases} \qquad j(x) := \begin{cases} 1, & \text{if } \mathrm{JS}_N(x) = -1 \\ 0, & \text{if } \mathrm{JS}_N(x) = 0 \text{ or } 1 \end{cases}$$

$$\mathrm{JS}_N(r) = -1, \quad \mathrm{JS}_N(s) = 1, \quad s \text{ a quadratic non-residue}$$

To learn $\mathrm{JS}_N(x)$:

$$\mathrm{JS}_N(f_{r,s,N}(x)) = \mathrm{JS}_N(r^{j(x)}) = \mathrm{JS}_N(x).$$

To learn the sign of x:

$$f_{r,s,N}(x) \cdot r^{-j(x)} = x^2 \cdot s^{h(x)} \text{ is a quadratic residue} \Leftrightarrow h(x) = 0.$$

Inverting injective functions

$$f_{r,s,N}(x) = x^2 \cdot r^{j(x)} \cdot s^{h(x)} \bmod N$$

Given the factorization of N, we can invert $f_{r,s,N}(x)$ by:

1. Compute $\mathrm{JS}_N(f_{r,s,N}(x))$ to learn $\mathrm{JS}_N(x)$.

Inverting injective functions

$$f_{r,s,N}(x) = x^2 \cdot r^{j(x)} \cdot s^{h(x)} \bmod N$$

Given the factorization of N, we can invert $f_{r,s,N}(x)$ by:

1. Compute $\mathrm{JS}_N(f_{r,s,N}(x))$ to learn $\mathrm{JS}_N(x)$.
2. Multiply by $r^{-j(x)}$.

Inverting injective functions

$$f_{r,s,N}(x) \cdot r^{-j(x)} = x^2 \cdot s^{h(x)} \bmod N$$

Given the factorization of N, we can invert $f_{r,s,N}(x)$ by:

1. Compute $\mathrm{JS}_N(f_{r,s,N}(x))$ to learn $\mathrm{JS}_N(x)$.
2. Multiply by $r^{-j(x)}$.
3. Determine whether result is a quadratic residue to learn $h(x)$.

Inverting injective functions

$$f_{r,s,N}(x) \cdot r^{-j(x)} = x^2 \cdot s^{h(x)} \bmod N$$

Given the factorization of N, we can invert $f_{r,s,N}(x)$ by:

1. Compute $\mathrm{JS}_N(f_{r,s,N}(x))$ to learn $\mathrm{JS}_N(x)$.
2. Multiply by $r^{-j(x)}$.
3. Determine whether result is a quadratic residue to learn $h(x)$.
4. Multiply by $s^{-h(x)}$.

Inverting injective functions

$$f_{r,s,N}(x) \cdot r^{-j(x)} \cdot s^{-h(x)} = x^2 \bmod N$$

Given the factorization of N, we can invert $f_{r,s,N}(x)$ by:

1. Compute $\mathrm{JS}_N(f_{r,s,N}(x))$ to learn $\mathrm{JS}_N(x)$.
2. Multiply by $r^{-j(x)}$.
3. Determine whether result is a quadratic residue to learn $h(x)$.
4. Multiply by $s^{-h(x)}$.
5. Compute four square roots and output the one that matches $h(x), j(x)$.

Creating lossy functions

$$f_{r,s,N}(x) = x^2 \cdot r^{j(x)} \cdot s^{h(x)} \bmod N$$

$$\mathrm{JS}_N(r) = -1, \quad \mathrm{JS}_N(s) = 1, \quad s \text{ a quadratic non-residue}$$

To create a lossy function, choose s with $\mathrm{JS}_N(s) = 1$ and s a *quadratic residue*.

- Function $f_{r,s,N}(x)$ is now 2-to-1 (one bit of lossiness) — loses information about the sign of x.
- Lossy functions $f_{r,s,N}$ are indistinguishable from injective functions $f_{r,s,N}$ under *quadratic residuosity assumption.*

Extending the system

Functions with index-independent domains (necessary for some applications):

- Can achieve $\log_2(4/3)$ bits of lossiness.

Using eth power residuosity assumption for $e > 2$:

- Can achieve $\log_2(e)$ bits of lossiness for $e < N^{1/4}$ with small prime factors.
- Inversion uses *Eisenstein reciprocity* in number fields.

See full version at `http://eprint.iacr.org/2009/590` for details.

Outline

1. Lossy and Correlation-Secure Trapdoor Functions
2. LTDFs from Quadratic Residuosity
3. LTDFs from d-Linear assumptions

A motivating observation

View $x \in \{0,1\}^n$ as a length-n vector $\vec{x}$.
Let M be an $n \times n$ matrix over $\mathbb{F}_p$. Consider

$$f_M(\vec{x}) := M \cdot \vec{x} \in \mathbb{F}_p^n$$

This function can be lossy or injective!

1. Injective:
   - If M has rank n, then $f_M(x)$ is invertible.
   - Need to know M^{-1} to invert.
2. Lossy:
   - If M has rank d, then $f_M(x)$ has image of size at most p^d.
   - If $p^d < 2^n$ then image is smaller than domain.

But we can easily distinguish these two cases by computing $rank(M)$.

A motivating observation

View $x \in \{0,1\}^n$ as a length-n vector $\vec{x}$.
Let M be an $n \times n$ matrix over $\mathbb{F}_p$. Consider

$$f_M(\vec{x}) := M \cdot \vec{x} \in \mathbb{F}_p^n$$

This function can be lossy or injective!

1. Injective:
   - If M has rank n, then $f_M(x)$ is invertible.
   - Need to know M^{-1} to invert.
2. Lossy:
   - If M has rank d, then $f_M(x)$ has image of size at most p^d.
   - If $p^d < 2^n$ then image is smaller than domain.

But we can easily distinguish these two cases by computing $rank(M)$.

A motivating observation

View $x \in \{0,1\}^n$ as a length-n vector $\vec{x}$.
Let M be an $n \times n$ matrix over $\mathbb{F}_p$. Consider

$$f_M(\vec{x}) := M \cdot \vec{x} \in \mathbb{F}_p^n$$

This function can be lossy or injective!

1. Injective:
   - If M has rank n, then $f_M(x)$ is invertible.
   - Need to know M^{-1} to invert.
2. Lossy:
   - If M has rank d, then $f_M(x)$ has image of size at most p^d.
   - If $p^d < 2^n$ then image is smaller than domain.

But we can easily distinguish these two cases by computing $rank(M)$.

A motivating observation

View $x \in \{0,1\}^n$ as a length-n vector $\vec{x}$.
Let M be an $n \times n$ matrix over $\mathbb{F}_p$. Consider

$$f_M(\vec{x}) := M \cdot \vec{x} \in \mathbb{F}_p^n$$

This function can be lossy or injective!

1. Injective:
   - If M has rank n, then $f_M(x)$ is invertible.
   - Need to know M^{-1} to invert.
2. Lossy:
   - If M has rank d, then $f_M(x)$ has image of size at most p^d.
   - If $p^d < 2^n$ then image is smaller than domain.

But we can easily distinguish these two cases by computing $rank(M)$.

Making injective and lossy functions indistinguishable

Idea: encode M in *exponent* of a group where discrete log is hard.

$$\mathbb{G} = \text{ group of order p}, \quad g \text{ a generator}, \quad M = (m_{ij}) \in \mathbb{F}_p^{n \times n}$$

Function description is $g^M := (g^{m_{ij}}) \in \mathbb{G}^{n \times n}$; trapdoor is M^{-1}.

- Evaluation:
  - $f_{g^M}(\vec{x}) := g^{M \cdot \vec{x}} \in \mathbb{G}^n$.
  - Can be easily computed from g^M and $\vec{x}$.
- Inversion (if M is full rank):
  1. Apply M^{-1} in exponent to recover $g^{\vec{x}} \in \mathbb{G}^n$ (also easy).
  2. Take discrete logs to recover $\vec{x}$ (easy since $\vec{x} \in \{0, 1\}^n$).

Making injective and lossy functions indistinguishable

Idea: encode M in *exponent* of a group where discrete log is hard.

$$\mathbb{G} = \text{ group of order p}, \quad g \text{ a generator}, \quad M = (m_{ij}) \in \mathbb{F}_p^{n\times n}$$

Function description is $g^M := (g^{m_{ij}}) \in \mathbb{G}^{n\times n}$; trapdoor is M^{-1}.

- Evaluation:
  - $f_{g^M}(\vec{x}) := g^{M\cdot\vec{x}} \in \mathbb{G}^n$.
  - Can be easily computed from g^M and $\vec{x}$.
- Inversion (if M is full rank):
  1. Apply M^{-1} in exponent to recover $g^{\vec{x}} \in \mathbb{G}^n$ (also easy).
  2. Take discrete logs to recover $\vec{x}$ (easy since $\vec{x} \in \{0,1\}^n$).

Security

Theorem [BHHO08,NS09]: if *d-linear assumption* holds in $\mathbb{G}$, then

$$\{g^M : rank(M) = n\} \quad \text{and} \quad \{g^M : rank(M) = d\}$$

are computationally indistinguishable.

- d-Linear assumption: generalization of DDH that may hold in groups with a d-linear map [BBS04,HK07,S07].
- $d = 1$ is DDH; $d = 2$ is "decision linear."

When $rank(M) = d$, amount of lossiness is $n - d \log_2 p$ bits.

Observations and extensions

1. Simplifies and generalizes [PW08] ElGamal-based construction
    - Save space by using random M instead of $M \in \{0,1\}^{n \times n}$.
    - Avoid generalized ElGamal encryption (d times as large).
2. Can choose parameters to achieve varying amounts of lossiness.
3. Admits an “all-but-one” generalization (DDH only).
    - Needed for [PW08] construction of CCA-secure encryption.
    - Details in full paper.

Conclusions

We showed constructions of lossy trapdoor functions based on *quadratic residuosity* and *d-Linear* assumptions.

- Also in paper: *composite residuosity* (Paillier) assumption, correlation-security from *syndrome decoding*.

Expanding our "library" of LTDFs expands the methods we have for creating cryptosystems in a simple and modular way.

Conclusions

We showed constructions of lossy trapdoor functions based on *quadratic residuosity* and *d-Linear* assumptions.

- Also in paper: *composite residuosity* (Paillier) assumption, correlation-security from *syndrome decoding*.

Expanding our "library" of LTDFs expands the methods we have for creating cryptosystems in a simple and modular way.

Thank you!