Lossy Trapdoor Functions

Giacomo Fenzi

ETH Zurich

22 April 2021

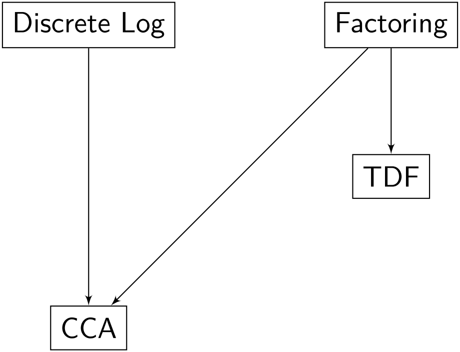
Motivation

- Trapdoor Functions are basic primitive, but hard to instantiate
- CCA Security from factoring and discrete log but not lattices

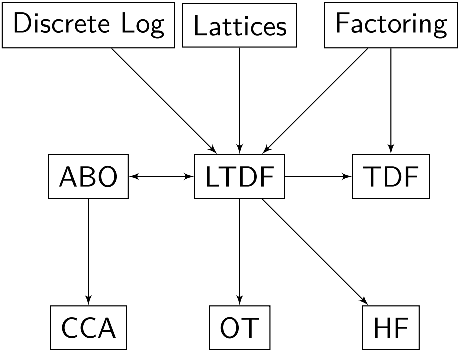
Results

- Introduce Lossy Trapdoor Functions (LTDFs)
- Realize LTDFs from factoring, discrete log *and* lattices
- Show LDTFs imply TDFs
- Black box construction of CCA-secure (witness recovering) cryptosystems, collision-resistant hash functions and oblivious transfer protocols.

Connections



Connections



Notation and Entropy

- λ is the security parameter, and we will abbreviate $n(\lambda) = \text{poly}(\lambda)$ as simply n
- $f(-)$ denotes the function taking $x \mapsto f(x)$
- Write $H_\infty(X)$ for the min-entropy of X. This corresponds to the optimal probability of guessing X.
- We let $\widetilde{H}_\infty(X|Y)$ be the average min-entropy of X conditioned on Y. This corresponds to the optimal probability of guessing X knowing Y.
- We use the following lemma, if Y takes at most 2^r values then:

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

Trapdoor Functions

Informally, a trapdoor function is family of functions that are hard to invert without access to some additional information called a trapdoor

Definition

A trapdoor function consists of three PPT algorithms (S, F, F^{-1}) such that:

- *Easy to sample and invert with trapdoor.* $S(1^\lambda) \to (s, t)$ such that $F(s, -)$ is an injective function on $\{0, 1\}^n$ and $F^{-1}(t, -)$ is its inverse
- *Hard to invert without.* For *any* PPT inverter $\mathcal{A}$ we have that $\mathcal{A}(1^\lambda, s, F(s, x))$ outputs x with negligible probability.

Example of Trapdoor

RSA Encryption! In trapdoor form:

- $S(1^\lambda)$ generates N, e, d as in RSA, set $s := (N, e)$ and $t := (d)$ and returns (s, t)
- $F(s, x)$ computes $x^e \mod N$
- $F^{-1}(t, c)$ computes $c^d \mod N$

Composite Residuosity

- $S(1^\lambda)$ generates $N = pq$ as a product of large primes, select g suitably, $s := (N, g)$, $t := (p, q)$
- $F(s, x)$ splits $x = m_1 + Nm_2$ and returns $g^{m_1} m_2^N \mod N^2$
- $F^{-1}(t, c)$ decrypts using the factorization to compute Carmichael function

Lossy Trapdoors

Informally, you either get an injective trapdoor or a 'lossy' function, and *cannot tell which is which*

Definition

A (n, k)-lossy trapdoor function consists of three PPT algorithms (S, F, F^{-1}). We denote $S_{inj}(-) \triangleq S(-, 0)$ and $S_{lossy}(-) \triangleq S(-, 1)$.

- *Outputs of S_{inj} are easy to compute and easy to invert with trapdoor.* $S_{inj}(1^\lambda) \to (s, t)$ s.t. that $F(s, -)$, $F^{-1}(t, -)$ are in the trapdoor case
- *Outputs of S_{lossy} are easy to compute.* $S_{lossy}(1^\lambda) \to (s, \perp)$ s.t. $F(s, -)$ is a function on $\{0, 1\}^n$ with image size at most 2^{n-k}.
- The first outputs of $S_{inj}(1^\lambda)$ and $S_{lossy}(1^\lambda)$ are computationally indistinguishable.

Subleties

- The definition really relates to a collection of lossy trapdoor functions.
- $k \triangleq k(\lambda) = \text{poly}(\lambda) \leq n$ is a parameter that represents how 'lossy' the collection is.
- We also write $r \triangleq n - k = \text{poly}(\lambda)$ as the *residual leakage*.
- No hardness requirement on inverting outputs of S_{inj}
- Requirements are too strict in lattices, leads to *almost-always* lossy functions.

All-But-One TDFs

Intuition: Most branches are trapdoors, except one which is lossy. You cannot tell which one it is.

Definition

An (n, k)-ABO TDF is a triple of PPT algorithms S, F, F^{-1} such that:

- $S(1^\lambda, b^*) \to (s, t)$ as before
- For any $b \neq b^*$, $F(s, b, -)$ $F^{-1}(t, b, -)$ are as in the previous definition.
- $F(s, b^*, -)$ is a lossy function as before
- For any b, b' the first outputs of $S(1^\lambda, b)$, $S(1^\lambda, b')$ are computationally indistinguishable.

ABO $\equiv$ LTDF

- ABOs and LTDFs are equivalent.
- ABO $\implies$ LTDF. Take ABO on $\{0,1\}$ and evaluate always on one of the branches, but switch lossy branch on generation.
- LTDF $\implies$ ABO. Generate an ABO on $\{0,1\}$ by having $s = (s_0, s_1)$ where one of the two is lossy, and evaluation by using s_b
- Finally, we can extend ABOs on $\{0,1\}$ to ABOs on $\{0,1\}^\ell$ at the cost of having residual leakage ℓr. The idea is, for lossy branch $b^* \in \{0,1\}^\ell$, generate ℓ ABOs each with the i-th having lossy branch b_i^*.

LTDF $\implies$ TDF

- Completeness: Use the injective functions generated by S_{inj}.
- Soundness: We cannot (information theoretically) invert the lossy branch, so if we could invert the injective trapdoors we could distinguish outputs of S_{inj}, S_{lossy}, contradicting LDTFness.
- Formally, let $\mathcal{A}$ be an inverter. We build $\mathcal{D}$

$$\underline{\mathcal{D}^{\mathcal{A}}(s)}$$

$$x \leftarrow_{\$} \{0,1\}^n$$
$$y = F(s, x)$$
$$x' = \mathcal{A}(s, y)$$
$$\textbf{return } x = x'$$

We analyze this in the next slide

LTDF $\implies$ TDF

Note that if s is generated by S_{inj} then with some non negligible probability we have that $\mathcal{A}$ succeeds and $\mathcal{D}$ succeeds whenever $\mathcal{A}$ does.
Instead, if s is generated by S_{lossy} even an unbounded adversary would have best possible probability given by $2^{-\widetilde{H}_\infty(x|s,F(s,x))}$. But note that $F(s,-)$ takes at most 2^r values and so by the previous lemma $\widetilde{H}_\infty(x|s,F(s,x)) \geq H_\infty(x|s) - r = n - (n-k) = k$. So the probability is bounded by 2^{-k} and as such is negligible.
From the above it follows that $\mathcal{D}$ will win the distinguishing game with non negligible probability.

LTDF $\implies$ CCA

We will have some requirements primitives[1]. We note that our cryptosystem will have message space $\{0,1\}^\ell$.

- We have $\Sigma = (\mathrm{Gen}, \mathrm{Sign}, \mathrm{Vfy})$ a strongly unforgeable one-time signature scheme. We require that the public keys are in $\{0,1\}^v$.
- $F = (S_{ltdf}, F_{ltdf}, F_{ltdf}^{-1})$ is a (n,k)-lossy trapdoor function.
- $G = (S_{abo}, F_{abo}, F_{abo}^{-1})$ is a (n,k')-ABO trapdoor function with branch space $\{0,1\}^v$.
- $\mathcal{H}$ is a collection of pairwise independent hash functions $\{0,1\}^n \to \{0,1\}^\ell$.
- We require that $k + k' \geq n + \kappa$ for some $\kappa = \omega(\log n)$ and that $\ell \leq \kappa - 2\lg(1/\epsilon)$ from $\epsilon = \mathrm{negl}(\lambda)$

[1] All of these reduce to LTDFs

LTDF $\implies$ CCA

$\mathcal{G}(1^\lambda)$	$\mathcal{E}(pk, m)$	$\mathcal{D}(sk, c)$
$(s, t) \leftarrow S_{inj}(1^\lambda)$	$(vk, sk_\sigma) = \mathrm{Gen}(1^\lambda)$	**if** $\neg\mathrm{Vfy}(\mathrm{vk}, (\mathrm{c_i})_{\mathrm{i}=1}^3, \sigma)$
$(s', t') \leftarrow S_{abo}(1^\lambda, 0^v)$	$x \leftarrow_\$ \{0, 1\}^n$	**return** $\perp$
$h \leftarrow_\$ \mathcal{H}$	$c_1 = F_{ltdf}(s, x)$	**fi**
$pk := (s, s', h)$	$c_2 = G_{abo}(s, vk, x)$	$x = F^{-1}(t, c_1)$
$sk := (t, t', pk)$	$c_3 = m \oplus h(x)$	**if** $c_1 \neq F_{ltdf}(s, x) \vee$
return (pk, sk)	$\omega \leftarrow \mathrm{Sign}(sk_\sigma, (c_i)_{i=1}^3)$	$c_2 \neq G_{abo}(s, vk, x)$
	return $(vk, c_1, c_2, c_3, \sigma)$	**return** $\perp$
		fi
		return $c_3 \oplus h(x)$

LTDF $\implies$ CCA

$\underline{\text{Setup}(\lambda)}$	$\underline{\text{EncO}(m_0, m_1)}$	$\underline{\text{DecO}(c^*)}$
$b \leftarrow_\$ \{0,1\}$	$c \rightarrow \mathcal{E}(pk, m_b)$	$\mathbf{if}\ c^* \in \mathcal{T}_{enc}$
$\mathcal{T}_{enc} = \emptyset$	$\mathcal{T}_{enc} \coloneqq \mathcal{T}_{enc} \cup \{c\}$	$\mathbf{return}\ \perp$
$pk, sk \rightarrow \mathcal{G}(\lambda)$	$\mathbf{return}\ c$	$\mathbf{fi}$
$\mathbf{return}\ pk$		$\mathbf{return}\ \mathcal{D}(sk, c^*)$