The Magic of ELFs

Mark Zhandry
Massachusetts Institute of Technology, USA
mzhandry@gmail.com

Abstract

We introduce the notion of an *Extremely Lossy Function* (ELF). An ELF is a family of functions with an image size that is tunable anywhere from injective to having a polynomial-sized image. Moreover, for any efficient adversary, for a sufficiently large polynomial r (necessarily chosen to be larger than the running time of the adversary), the adversary cannot distinguish the injective case from the case of image size r.

We develop a handful of techniques for using ELFs, and show that such extreme lossiness is useful for instantiating random oracles in several settings. In particular, we show how to use ELFs to build secure point function obfuscation with auxiliary input, as well as polynomially-many hardcore bits for any one-way function. Such applications were previously known from strong knowledge assumptions — for example polynomially-many hardcore bits were only know from differing inputs obfuscation, a notion whose plausibility has been seriously challenged. We also use ELFs to build a simple hash function with *output intractability*, a new notion we define that may be useful for generating common reference strings.

Next, we give a construction of ELFs relying on the *exponential* hardness of the decisional Diffie-Hellman problem, which is plausible in pairing-based groups. Combining with the applications above, our work gives several practical constructions relying on qualitatively different — and arguably better — assumptions than prior works.

1 Introduction

Hash functions are a ubiquitous tool in cryptography: they are used for password verification, proofs of work, and are central to a variety of cryptographic algorithms including efficient digital signatures and encryption schemes.

Unfortunately, formal justifications of many of the uses of hash functions have been elusive. The trouble stems from the difficulty of even defining what security properties a hash function should satisfy. On one extreme, a hash function can be assumed to have standard security properties such as one-wayness or collision resistance, which are useless for most of the applications above. On the other extreme, a hash function can be modeled as a truly random function, where it is assumed that an adversary only has black-box access. In the so-called random oracle model (ROM) [BR93], all of the above applications are secure. However, random oracles clearly do not exist and moreover provably cannot be replaced by any concrete hash function [CGH98]. In this light, it is natural to ask:

What are useful properties of random oracles
that can be realized by real-world hash functions.

Some attempts have been made to answer this question; however, many such attempts have serious limitations. For example Canetti, Goldreich, and Halevi [CGH98] propose the notion of *correlation intractability* as a specific feature of random oracles that could potentially have a standard model instantiation. However, they show that for some parameter settings such standard model hash functions cannot exist. The only known positive example [CCR16] relies on extremely strong cryptographic assumptions such as general-purpose obfuscation. For another example, Bellare, Hoang, and Keelveedhi [BHK13] define a security property for hash functions called Universal Computational Extractors (UCE), and show that hash functions with UCE security suffice for several uses of the random oracle model. While UCE's present an important step toward understanding which hash function properties might be achievable and which are not, UCE's have several limitations. For example, the formal definition of a UCE is somewhat complicated to even define. Moreover, UCE is not a single property, but a family or "framework" of assumptions. The most general form of UCE is trivially unattainable, and some of the natural restricted classes of UCE have been challenged [BFM14, BST16]. Therefore, it is unclear which versions of UCE should be trusted and which untrusted.

Similar weaknesses have been shown for other strong assumptions that can be cast as families of assumptions or as knowledge/extracting assumptions, such as differing inputs obfuscation (diO) [BCP14, ABG$^+$13, GGHW14]. These weakness are part of a general pattern for strong assumptions such as UCE and diO that are not specified by a cryptographic game. In particular, these assumptions do not meet standard notions of falsifiability ([Nao03, GW11]), and are not *complexity assumptions* in the sense of Goldwasser and Kalai [GK16]. (We stress that such knowledge/extracting/framework assumptions are desirable as security *properties*. However, in order to trust that the property actually holds, it should be derived from a "nice" and trusted assumption.) Therefore, an important question in this space is the following:

Are there primitives with "nice" (e.g. simple, well-established, game-based, falsifiable, complexity assumption, etc) security properties that can be used to build hash functions suitable for instantiating random oracles for many protocols.

1.1 Our Work

Our Random Oracle Targets. In this work, we aim to base several applications of random oracles on concrete, "nice" assumptions with relatively simple instantiations.

- **Boosting selective to adaptive security.** A trivial application of random oracles is to boost selective to adaptive security in the case of signatures and identity-based encryption. This is done by first hashing the message/identity with the random oracle before signing/generating secret keys. There has been no standard-model primitive proposed that allows for this conversion.

- **Password hashing.** Another common use of hash functions is to securely store a password in "encrypted" form, allowing for the password to be verified, but hiding the actual password. This use case can be captured by the notion of *point obfuscation* (PO). In the case that there may be side information about the password, we have the notion of *auxiliary input* point obfuscation (AIPO). The only prior constructions of AIPO [Can97, BP12] rely on very strong knowledge assumptions. The first is Canetti's [Can97] strong knowledge variant of the decisional Diffie Hellman (DHH) assumption, whose plausibility has been called into question by a recent work showing it is incompatible with the existance of certain strong forms of obfuscation [BST16]. The second is a strong knowledge assumption about one-way permutations due to Bitansky and Paneth [BP12], which is a strengthening of Wee's strong one-way permutation assumption [Wee05]. To the best of our knowledge, the only known ways to instantiate the [BP12] assumption is to make the tautological assumption that a particular one-way permutation is secure. For reasons mentioned above, such tautological knowledge assumptions are generally considered undesirable in cryptography.

- **Generating system parameters.** A natural use case of hash functions is for generating common random strings (crs) in a trusted manner. More specifically, suppose a (potentially untrusted) authority is generating a crs for some protocol. Unfortunately, such a crs may admit a "trapdoor" that allows for breaking whatever protocol is using it (Dual_EC_DRBG is a prominent example of this). In order to ensure to untrusting parties that no trapdoor is known, the authority will generate the crs as an output of the hash function on some input. The authority may have some flexibility in choosing the input; we wish to guarantee that it cannot find an input such that it also knows a trapdoor for the corresponding output. In the random oracle model, this methodology is sound: the authority cannot choose an input so that it knows the trapdoor for the output. However, standard notions of security for hash functions do not preclude the possibility of the adversary choosing a strange input the the hash function so that it knows a trapdoor for the output. We propose (Section 5) the notion of *output intractability* as a standard-model security notion that captures this use case. Output intractability is related to, but incomparable with, the notion of correlation intractability mentioned above. As an assumption, our notion of output intractability takes the form as a knowledge assumption on hash functions; no construction based on "nice" assumptions is known.

- **Hardcore bits for any one-way function.** A random oracle serves as a good way to extract many hardcore bits for any one-way function. This fact gives rise to a simple public-key encryption scheme from trapdoor permutations. Unfortunately, for general one-way functions, the only known way to extract more than a logarithmic number of hardcore bits is to use very strong (and questionable [GGHW14]) knowledge assumptions: differing inputs obfuscation [BST14] or extractable witness PRFs [Zha16]. In the case of one-way *permutations*, Bellare, Stepanovs, and Tessaro [BST14] show that the weaker assumption of *indistiguishability* obfuscation (iO) suffices. While weaker that diO, iO is still one of the strongest assumptions made in cryptography. Either way, the forms of obfuscation required are also completely impractical [AHKM14]. Moreover, prior constructions required the randomness used to sample the hardcore function to be kept secret.

- **Instantiating Full Domain Hash (FDH) signatures.** Finally, we consider using random oracles to instantiate the Full Domain Hash (FDH) protocol transforming trapdoor permutations into signatures. Hohenberger, Sahai, and Waters [HSW14] show that (indistinguishability) obfuscating a (puncturable) pseudorandom function *composed with the permutation* is sufficient for FDH signatures. However, their proof has two important limitations. First, the resulting signature scheme is only selectively secure. Second, the instantiation depends on the particular trapdoor permutation used, as well as the public key of the signer. Thus, each signer needs a separate hash function, which needs to be appended to the signer's public keys. To use their protocol, everyone will therefore need to publish new keys.

Our approach. We take a novel approach to addressing the questions above. We isolate a (generally ignored) property of random oracles, namely that random oracles are indistinguishable from functions that are extremely lossy. More precisely, the following is possible in the random oracle model. Given any polynomial time oracle adversary $\mathcal{A}$ and a non-negligible probability ϵ, we can choose the oracle such that (1) the image size of the oracle is a polynomial r, and (2) $\mathcal{A}$ cannot tell the difference between such a lossy oracle and a truly random oracle, except with advantage smaller than ϵ. Note that the tuning of the image size must be done with knowledge of the adversary's running time — an adversary running in time $O(\sqrt{r})$ can with high probability find a collision, thereby distinguishing the lossy function from a truly random oracle. However, by setting $\sqrt{r}$ to be much larger than the adversary's running time, the probability of finding a collision diminishes. We stress that any protocol would still use a truly random oracle and hence not depend on the adversary; the image size tuning would only appear in the security proof. Our observation of this property is inspired by prior works of Boneh and Zhandry [Zha12, BZ13], who use it for the entirely different goal of giving security proofs in the so-called *quantum* random oracle model (random oracle instantiation was not a goal nor accomplishment of these prior works).

We next propose the notion of an *Extremely Lossy Function (ELF)* as a standard-model primitive that captures this tunable image size property. The definition is related to the notion of a lossy *trapdoor* function due to Peikert and Waters [PW08], with two important differences: we do not need any trapdoor, giving hope that ELFs could be constructed from symmetric primitives. On the other hand, we need the functions to be much, much more lossy, as standard lossy functions still have exponential image size.

On the surface, extreme lossiness without a trapdoor does not appear incredibly useful, since many interesting applications of standard lossy functions (e.g. (CCA-secure) public key encryption) require a trapdoor. Indeed, using an ELF as a hash function directly does not solve any of the tasks outlined above. Perhaps surprisingly, we show that this extremely lossy property, in conjunction with other tools — usually pairwise independence — can in fact quite powerful, and we use this power to give new solutions to each of the tasks above. Our results are as follows:

- (Section 3) We give a practical construction of ELFs assuming the *exponential* hardness of the decisional Diffie-Hellman (DDH) problem. Our construction also achieves a public coin notion, which is useful for obtaining public coin hash functions in applications. The construction is based on the lossy trapdoor functions due to Peikert and Waters [PW08], though simpler since we do not need a trapdoor. While our ELFs are built from public key tools, we believe such tools are unnecessary and we leave as an interesting open question the problem of constructing ELFs from symmetric or generic tools.

  The exponential hardness of DDH is plausible on elliptic curve groups — despite over a decade of wide-spread use and cryptanalysis attempts, there are virtually no non-trivial attacks on most elliptic curve groups. In fact, the parameter settings for most real-world uses of the Diffie-Hellman problem are set assuming the Diffie-Hellman problem takes exponential time to solve. If our assumption turns out to be false, it would have significant ramifications for the internet, as it would suggest that parameters for many cryptosystems in practice are set too aggressively. It would therefore be quite surprising if DDH turned out to *not* be exponentially hard on elliptic curves. While not a true falsifiable assumption in the sense of Naor [Nao03] or Gentry and Wichs [GW11] due to the adversary being allowed to run in exponential time, the exponential DDH assumption is falsifiable in spirit and naturally fits within the complexity assumption framework of Goldwasser and Kalai [GK16].

- We give several different hash function instantiations based on ELFs ranging in complexity and the additional assumptions used. In doing so, we give new solutions to each of the problems above. Each construction uses the ELFs in different ways, and we develop new techniques for the analysis of these constructions. Thus we give an initial set of tools for using ELFs that we hope to be useful outside the immediate scope of this work.
  - The simplest instantiation is just to use an ELF itself as a hash function. Such a function can be used to generically boost selective security to adaptive security in signatures and identity-based encryption by first hashing the message/user identity (more details below).
  - (Section 4) The next simplest instantiation is to pre-compose the ELF with a pairwise independent hash function. This function gives rise to a simple (public coin) point function obfuscation (PO). Proving this requires a new variant of the "leftover hash lemma" [ILL89], which may be of independent interest.
  - (Section 5) A slightly more complicated instantiation is given by *post*-composing and ELF with a k-wise independent function. We show that this construction satisfies our notion of *output intractability*. It is moreover public coin.
  - (Section 6) We then give an even more complicated construction, though still using ELF's as the only underlying source of cryptographic hardness. We demonstrate that our construction is a pseudorandom generator attaining a very strong notion of leakage resilience for the seed. This property generalizes the strong variant of one-wayness of Bitansky and Paneth [BP12]. Our construction therefore shows how to instantiate the knowledge properties conjectured in their work using a more traditional-style assumption.

    An immediate consequences of our generator requirement is a (public coin) point function obfuscation that is secure even in the presence of auxiliary information (AIPO), which was previously known from either *permutations* satisfying [BP12]'s one-wayness requirement (our function is *not* a permutation), or from Canetti's strong knowledge variant of DDH [Can97, BP12]. Our AIPO construction is qualitatively very different from these existing constructions, and when plugging in our ELF construction, again relies on just exponential DDH.

    Our generator also immediately gives a family of (public coin) hardcore functions of arbitrary stretch for any one-way function. Unlike the previous obfuscation-based solutions, our is both practical, and public coin, and ultimately based on a well-studied game-based assumption.

    Our analysis also demonstrates that our ELF-based function can be used in a standard random oracle public key encryption protocol [BR93].
  - (Section 7) Finally, we give an instantiation that involves obfuscating the composition of an ELF and a (puncturable) pseudorandom function (but not the permutation) using an indistinguishability obfuscator. Since we use obfuscation as in Hohenberger, Sahai, and Waters' [HSW14] scheme, this construction is still completely impractical and therefore currently only of theoretical interest. We show that our construction can be used in the FDH protocol, solving the limitations in [HSW14]. In particular, by composing with an ELF, we immediately get adaptive security as observed above. Our construction is moreover independent of the permutation (except for the size of the circuit computing it), and is also independent of the signer's public key. Thus, our instantiation is universal and

one instantiation can be used by any signer, even using existing keys. Similar to [HSW14], this construction is still required to be secret coin, even if the underlying ELF is public coin.

Warm up: generically boosting selective to adaptive security. To give a sense for our techniques, we show how ELFs can be used to generically boost selective to adaptive security in signatures and identity-based encryption. We demonstrate the case for signatures; the case for identity based encryption is almost identical.

Recall that in selective security for signatures, the adversary commits to a message m^* at the beginning of the experiment before seeing the public key. Then the adversary makes a polynomial q adaptive signing queries on messages $m_1, \dots, m_q$, receiving signatures $\sigma_1, \dots, \sigma_q$. It is required that $m_i \neq m^*$ for any i. Then, the adversary produces a forged signature σ^* on m^*, and security states that σ^* is with overwhelming probability invalid for any efficient adversary. Adaptive security, in contrast, allows the adversary to choose m^* potentially *after* the q adaptive queries.

We now convert selective to adaptive security using ELFs: first hash the message using the ELF, and then sign. Adaptive security is proved through a sequence of hybrids. The first is the standard adaptive security game above. Toward contradiction, suppose that the adversary runs in polynomial time t and succeeds in forging a signature on m^* with non-negligible probability ϵ. In the second hybrid, the ELF is selected to have polynomial image size r, where $r \geq 2q$ is chosen, say, so that no t-time adversary can distinguish between this ELF and an injective ELF, except with probability at most $\epsilon/2$. Thus, in this hybrid, the adversary still successfully forges with probability $\epsilon/2$.

In the next hybrid, at the beginning of the experiment, one of the r image points of the ELF, y^*, is chosen at random[1]. Then we abort the experiment if the adversary's chosen m^* does not hash to y^*; with probability $1/r$, we do not abort[2]. This abort condition is independent of the adversary's view, meaning that we do not abort, and the adversary successfully forges, with probability at least $\epsilon/2r$, which is non-negligible. Notice now that y^* can be chosen at the beginning of the experiment. This is sufficient for obtaining an adversary for the selective security of the original signature scheme.

1.2 Complexity Absorption

Not all assumptions are created equal, and it may be more reasonable to assume the sub-exponential or exponential hardness of an existing well-studied problem than to assume such hardness for new and untested problems. Moreover, there might be implementation issues (such as having to re-publish longer keys, see Section 7 for a setting where this could happen) that make assuming the sub-exponential hardness of certain primitives undesirable.

The result above can be seen as an instance of a more general task of *complexity absorption*, where an extra complexity-absorbing primitive (in our case, and ELF) is introduced into the protocol. The original building blocks of the protocol (the underlying signature/identity-based encryption in this case) can be reduced from (sub)exponential security to polynomial security. Meanwhile, the

[1]The ability to sample a random image point does not follow immediately from our basic ELF definition, though this can be done in our construction. If the ELF is regular, then this can be accomplished by sampling a random input to the ELF and then applying the ELF. More generally, if it is possible to efficiently enumerate all the image points, then randomly sampling an image point is easy. Of course, enumerating all the image points will take time at least r, which is larger than the running time of the adversary, but can still potentially be done efficiently.

[2]We also need to abort if any of the m_i do hash to y_i. It is straightforward to show that we still do not abort with probability at least $\frac{1}{2r}$.

complexity-absorbing primitive may still require exponential hardness as in our case, but hopefully such hardness is a reasonable assumption. Our hardcore function with arbitrary span can also be seen in this light: it is straightforward to extend Goldriech-Levin [GL89] to a hardcore function of polynomial span for exponentially-secure one-way functions. By introducing an ELF into the hardcore function, the ELF can absorb the complexity required of the one-way function, yielding a hardcore function for *any* one-way function, even one-way functions that are only polynomially secure. Similarly, our random oracle instantiation for Full Domain Hash can also be seen as an instance of complexity absorption.

Thus, our work can be seen as providing an initial set of tools and techniques for the task of complexity absorption that may be useful in other settings where some form of sub-exponential hardness is difficult or impossible to avoid. For example, Rao [Rao14] argues that any proof of adaptive security for multiparty non-interactive key exchange (NIKE) will likely incur an exponential loss. As all current multiparty NIKE protocols are built from multilinear maps or obfuscation, which in turn rely on new, untested (and in many cases broken) hardness assumptions, assuming the sub-exponential security of the underlying primitives to attain adaptive security is undesirablee. Hofheinz et al. [HJK$^+$14] give a construction in the random oracle model that only has a polynomial loss; our work gives hope that a standard model construction based on ELFs may be possible where the ELF is the only primitive that needs stronger than polynomial hardness.

For a more ambitious example, Garg, Gentry, Sahai, and Waters [GGSW13] argue that any proof of security of witness encryption and obfuscation will also likely incur an exponential loss. This suggests that at least one of the underlying assumptions (those on multilinear maps) will need to be sub-exponentially secure. An intriguing possibility is to use ELFs (or some other (sub)exponentially secure primitive with better-understood security) to absorb the complexity, yielding an obfuscation construction from a constant number of assumptions *without* assuming the sub-exponential hardness of multilinear maps.

1.3 Non-black box simulation

Our proofs inherently require knowledge of the adversary's running time (and success probability). Thus, they do not make black box use of the adversary. Yet, this is the only non-black box part of our proofs — the reduction does not need to know the description or internal workings of the adversary. This is similar to the case of Goldreich-Levin [GL89], where only the adversary's success probability is needed. Thus our reductions are very nearly black box, while potentially giving means to circumvent black-box impossibility results. For example, proving the security of AIPO is known to require non-black box access to the adversary [Wee05, BP12], and yet our reduction proves the security of AIPO knowing only the adversary's running time and success probability. We leave it as an interesting open question to see if your techniques can be used to similarly circumvent other black box impossibility results.

2 Preliminaries

Given a distribution $\mathcal{D}$ over a set $\mathcal{X}$, define the support of $\mathcal{D}$, $\text{Supp}(\mathcal{D})$, to be the set of points in $\mathcal{X}$ that occur with non-zero probability. For any $x \in \mathcal{X}$, let $\Pr[\mathcal{D} = x]$ be the probability that $\mathcal{D}$ selects x. For any set $\mathcal{X}$, define $U_{\mathcal{X}}$ to be the uniform distribution on $\mathcal{X}$. Define the collision probability of

$\mathcal{D}$ to be

$$CP(\mathcal{D}) = \Pr[x_1 = x_2 : x_1, x_2 \leftarrow \mathcal{D}] = \sum_{x \in \mathcal{X}} \Pr[\mathcal{D} = x]^2 \ .$$

Given two distributions $\mathcal{D}_1, \mathcal{D}_2$, define the statistical distance between $\mathcal{D}_1$ and $\mathcal{D}_2$ to be

$$\Delta(\mathcal{D}_1, \mathcal{D}_2) = \frac{1}{2} \sum_{x \in \mathcal{X}} \left| \Pr[\mathcal{D}_1 = x] - \Pr[\mathcal{D}_2 = x] \right| \ .$$

Suppose $\mathrm{Supp}(\mathcal{D}_1) \subseteq \mathrm{Supp}(\mathcal{D}_2)$. Define the Rényi Divergence between $\mathcal{D}_1$ and $\mathcal{D}_2$ to be

$$RD(\mathcal{D}_1, \mathcal{D}_2) = \sum_{x \in \sup(\mathcal{D}_1)} \frac{\Pr[\mathcal{D}_1 = x]^2}{\Pr[\mathcal{D}_2 = x]} \ ^{3} \ .$$

The Rényi divergence is related to the statistical distance via the following lemma:

Lemma 2.1. *For any distributions $\mathcal{D}_1, \mathcal{D}_2$ over a set $\mathcal{Z}$ such that* $\mathrm{Supp}(\mathcal{D}_1) \subseteq \mathrm{Supp}(\mathcal{D}_2)$*,*

$$\Delta(\mathcal{D}_1, \mathcal{D}_2) \leq \frac{1}{2} \sqrt{RD(\mathcal{D}_1, \mathcal{D}_2) - 1} \ .$$

Consider a distribution $\mathcal{H}$ over the set of functions $h : \mathcal{X} \to \mathcal{Y}$. We say that $\mathcal{H}$ is *pairwise independent* if, for any $x_1 \neq x_2 \in \mathcal{X}$, the random variables $\mathcal{H}(x_1)$ and $\mathcal{H}(x_2)$ are independent and identically distributed, though not necessarily uniform. Similarly define k-wise independence. We say that $\mathcal{H}$ has *output distribution* $\mathcal{D}$ if for all x, the random variable $\mathcal{H}(x)$ is identical to $\mathcal{D}$. Finally, we say that $\mathcal{H}$ is *uniform* if it has output distribution $U_{\mathcal{Y}}$[4]. We will sometimes abuse notation and say that a function h is a pairwise independent function (resp. uniform) if h is drawn from a pairwise independent (resp. uniform) distribution of functions.

We will say that a (potentially probabilistic) algorithm $\mathcal{A}$ outputting a bit b distinguishes two distributions $\mathcal{D}_0, \mathcal{D}_1$ with advantage ϵ if the random variables $\mathcal{A}(\mathcal{D}_0)$ and $\mathcal{A}(\mathcal{D}_1)$ have ϵ statistical distance.

Unless otherwise stated, all cryptographic protocols will implicitly take a security parameter λ as input. Moreover, any sets (such as message spaces, ciphertext spaces, etc) will be implicitly indexed by λ, unless otherwise stated. In this context, when we say that an adversary is efficient, we mean its running time is polynomial in λ. A non-negative function ϵ is *negligible* if it is smaller than any inverse polynomial. When discussing cryptographic protocols, we say that a probability of an event or advantage of an adversary is negligible if it is negligible in λ. Two distributions $\mathcal{D}_0, \mathcal{D}_1$ (implicitly parameterized by λ) are computationally indistinguishable if any efficient algorithm has only negligible distinguishing advantage.

2.1 A New Leftover Hash Lemma

Here we prove a new variant of the leftover hash lemma which will be useful in some of our analysis, and may be of independent interest.

[3]Often, the Rényi Divergence is defined to be proportional to the logarithm of this quantity. For our purposes, this representation of the divergence will be more convenient.

[4]Note that the typical use of *pairwise independence* is equivalent to our notion of pairwise independence *plus* uniformity. For our purposes, it will be convenient to separate out the two properties.

Lemma 2.2. *Let H be a distribution on functions $h : \mathcal{X} \to \mathcal{Y}$ that is* pairwise independent *with output distribution $\mathcal{E}$, for some distribution $\mathcal{E}$ that is possibly non-uniform. Let $\mathcal{D}$ be an arbitrary distribution over $\mathcal{X}$. Then we have that*

$$\Delta(\ (H, H(\mathcal{D}))\ ,\ (H, \mathcal{E})\ ) \leq \frac{1}{2}\sqrt{CP(\mathcal{D})(\ |\mathrm{Supp}(\mathcal{E})| - 1\ )}$$

Proof. Our proof will proceed analogously to the proof of the original Leftover Hash Lemma [ILL89]. [ILL89] proceeds by bounding the collision probability of $(H, H(\mathcal{D}))$, and then showing that distributions with low collision probability must be close to uniform. In general, the collision probability of a distribution $\mathcal{F}$ is equivalent to the Rényi divergence $RD(\mathcal{F}, U)$ between $\mathcal{F}$ and the uniform distribution, up to an overall scaling factor. Therefore, [ILL89] can be equivalently proved by bounding the Rényi divergence, and then relating the divergence to statistical distance using Lemma 2.1. This is the view we use for our proof.

Given the following claim and Lemma 2.1, Lemma 2.2 follows:

Claim 2.3. $RD(\ (H, H(\mathcal{D}))\ ,\ (H, \mathcal{E})\ ) = 1 + CP(\mathcal{D})(|\mathrm{Supp}(\mathcal{E})| - 1)$

We now prove the claim. First,

$$RD(\ (H, H(\mathcal{D}))\ ,\ (H, \mathcal{E})\ ) = \sum_{h,y} \frac{\Pr[H = h \wedge h(\mathcal{D}) = y]^2}{Pr[H = h]Pr[\mathcal{E} = y]} = \sum_{y} \frac{\sum_h \Pr[H = h] \Pr[h(\mathcal{D}) = y]^2}{\Pr[\mathcal{E} = y]} \ .$$

Now, notice that $\Pr[h(\mathcal{D}) = y]^2 = \Pr[h(x_1) = y \wedge h(x_2) = y : x_1, x_2 \leftarrow \mathcal{D}]$, so that

$$\sum_h \Pr[H = h] \Pr[h(\mathcal{D}) = y]^2 = \Pr[H(x_1) = y \wedge H(x_2) = y : x_1, x_2 \leftarrow \mathcal{D}] \ .$$

Also, notice that $\Pr[E = y] = \Pr[H(x_1) = y]$ for any $x_1 \in \mathcal{X}$. Therefore, we can write

$$RD(\ (H, H(\mathcal{D}))\ ,\ (H, \mathcal{E})\ ) = \sum_{x_1, x_2} \Pr[X = x_1] \Pr[X = x_2] \sum_y \Pr[H(x_1) = y | H(x_2) = y] \ .$$

Now we divide into two cases. If $x_1 = x_2$, then $\Pr[H(x_1) = y|H(x_2) = y] = 1$, so $\sum_y \Pr[H(x_1) = y|H(x_2) = y] = |\mathrm{Supp}(\mathcal{E})|$. If $x_1 \neq x_2$, then by independence $\Pr[H(x_1) = y|H(x_2) = y] = \Pr[H(x_1) = y] = \Pr[\mathcal{E} = y]$, so $\sum_y \Pr[H(x_1) = y|H(x_2) = y] = \sum_y \Pr[\mathcal{E} = y] = 1$. Therefore,

$$\begin{aligned} RD(\ (H, H(\mathcal{D}))\ ,\ (H, \mathcal{E})\ ) &= \sum_x \Pr[\mathcal{D} = x]^2 |\mathrm{Supp}(E)| + \sum_{x_1 \neq x_2} \Pr[\mathcal{D} = x_1] \Pr[\mathcal{D} = x_2] \\ &= CP(\mathcal{D})|\mathrm{Supp}(E)| + (1 - CP(\mathcal{D})) = 1 + CP(\mathcal{D})(|\mathrm{Supp}(E)| - 1) \ . \end{aligned}$$

This completes the proof of the claim, completing the proof of Lemma 2.2. □

3 Extremely Lossy Functions

Here, we define our notion of *extremely lossy functions*, or ELFs. A standard lossy function [PW08] is intuitively a function family with two modes: an injective mode where the function is injective, and a lossy mode where the image size of the function is much smaller than the domain. The standard security requirement is that no polynomial-time adversary can distinguish the two modes[5].

An ELF is a lossy function with a much stronger security requirement. In the lossy mode, the image size can be taken to be a polynomial r. Clearly, such a lossy mode can be distinguished from injective by an adversary running in time $O(\sqrt{r})$ that simply evaluates the function on $\sqrt{r}$ inputs, looking for a collision. Therefore, we cannot have security against arbitrary polynomial-time attackers. Instead, we require security against r^c-time attackers, for some $c \leq 1/2$. Moreover, we require that r is actually tunable, and can be chosen based on the adversary in question. This means that for *any* polynomial time attacker, we can set the lossy function to have domain r for some polynomial r, and the lossy function will be indistinguishable from injective to that particular attacker (note that the honest protocol will always use the injective mode, and therefore will not depend on the adversary in any way).

Definition 3.1. An *extremely lossy function* (ELF) consists of an algorithm `ELF.Gen`. `ELF.Gen` which takes as input integers M and $r \in [M]$. There is no security parameter here; instead, $\log M$ acts as the security parameter. `ELF.Gen` outputs the description of a function $f : [M] \to [N]$ such that:

- f is computable in time polynomial in the bit-length of its input, namely $\log M$. The running time is independent of r.
- If $r = M$, then f is injective with overwhelming probability (in $\log M$).
- For all $r \in [M]$, $|f([M])| \leq r$ with overwhelming probability. That is, the function f has image size at most r.
- For any polynomial p and non-negligible function ϵ (in $\log M$), there is a polynomial q such that: for any adversary $\mathcal{A}$ running in time at most p, and any $r \in [q(\log M), M]$, we have that $\mathcal{A}$ distinguishes $\texttt{ELF.Gen}(M, M)$ from $\texttt{ELF.Gen}(M, r)$ with advantage less than ϵ. Intuitively, no polynomial-time adversary $\mathcal{A}$ can distinguish an injective from polynomial image size (where the polynomial size depends on the adversary's running time.).

For some applications, we will need an additional requirement for ELFs:

Definition 3.2. An ELF has an *efficiently enumerable* image space if, for any polynomial r, there is an efficient procedure that, given $f \leftarrow \texttt{ELF.Gen}(M, r)$, outputs the entire list of (at most r) image points of f with overwhelming probability.

We note that enumerating the image space for an ELF is equivalent to sampling a random image point. In one direction, having the list of image points makes it trivial to sample a random point. In the other, by sampling $\lambda r \log r$ image points independently at random, except with negligible probability in λ, the set of sampled points will contain every image point.

Definition 3.3. An ELF is *public coin* if the description of an injective mode f outputted by $\texttt{ELF.Gen}(M, M)$ is simply the random coins used by $\texttt{ELF.Gen}(M, M)$. The descriptions of lossy mode f's outputted by $\texttt{ELF.Gen}(M, r), r < M$ may (and in fact, must) be a more complicated function of the random coins.

[5][PW08] additionally require that, in the injective mode, there is a trapdoor that allows inverting the funciton. We will not need any such trapdoor

3.1 Constructing ELFs

Our construction is based on the DDH-based lossy *trapdoor* function of Peikert and Waters [PW08]. We stress that we do not need the trapdoor property of their construction, only the lossy property. Security will be based on the exponential hardness of the decisional Diffie-Hellman problem.

Definition 3.4. A *cryptographic group* consists of an algorithm `GroupGen` that takes as input a security parameter λ, and outputs the description of a cyclic group $\mathbb{G}$ of prime order $p \in [2^\lambda, 2 \times 2^\lambda)$, and a generator g for $\mathbb{G}$ such that:

- The group operation $\times : \mathbb{G}^2 \to \mathbb{G}$ can be computed in time polynomial in λ.
- Exponentiation by elements in $\mathbb{Z}_p$ can be carried out in time polynomial in λ. This follows from the efficient group operation procedure by repeated doubling and the fact that $\log p \leq \lambda + 1$.
- The representation of a group element h has size polynomial in λ. This also follows implicitly from the assumption that the group operation is efficient.

We now introduce some notation. For a matrix $\mathbf{A} \in \mathbb{Z}_p^{m\times n}$, we write $g^{\mathbf{A}} \in \mathbb{G}^{m\times n}$ to be the $m \times n$ matrix of group elements $g^{A_{i,j}}$. Similarly define $g^{\mathbf{w}}$ for a vector $\mathbf{w} \in \mathbb{Z}_p^n$. Given a matrix $\hat{\mathbf{A}} \in \mathbb{G}^{m\times n}$ of group elements and a vector $\mathbf{v} \in \mathbb{Z}_p^n$, define $\hat{\mathbf{A}} \cdot \mathbf{v}$ to be $\hat{\mathbf{w}} \in \mathbb{G}^m$ where $\hat{w}_i = \prod_{j=1}^n \hat{A}_{i,j}^{v_j}$. Using this notation, $(g^{\mathbf{A}}) \cdot \mathbf{v} = g^{\mathbf{A}\cdot\mathbf{v}}$. Therefore, the map $g^{\mathbf{A}}, \mathbf{v} \mapsto g^{\mathbf{A}\cdot\mathbf{v}}$ is efficiently computable.

Definition 3.5. The *exponential decisional Diffie Hellman* (eDDH) assumption on a cryptographic group specified by `GroupGen` holds if there is a polynomial $q(\cdot,\cdot)$ such that the following is true. For any time bound t and probability ϵ, let $\lambda = \log q(t, 1/\epsilon)$. Then for any adversary $\mathcal{A}$ running in time at most t, the following two distributions are indistinguishable, except with advantage at most ϵ:

$$(\mathbb{G}, g, g^a, g^b, g^c) : (\mathbb{G}, g, p) \leftarrow \texttt{GroupGen}(\lambda), a, b, c \leftarrow \mathbb{Z}_p \text{ and}$$
$$(\mathbb{G}, g, g^a, g^b, g^{ab}) : (\mathbb{G}, g, p) \leftarrow \texttt{GroupGen}(\lambda), a, b \leftarrow \mathbb{Z}_p$$

The following will help us achieve a public coin ELF.

Definition 3.6. A cryptographic group is *public coin* if the following holds:

- The "description" of $\mathbb{G}, g, p$ is just the random coins sampled by `GroupGen`.
- There is a (potentially redundant) efficiently computable representation of group elements in $\mathbb{G}$ as strings in $\{0,1\}^n$ such that (1) a random string in $\{0,1\}^n$ corresponds to a random element in $\mathbb{G}$, and (2) a random representation of a random element in $\mathbb{G}$ is a random string in $\{0,1\}^n$.

A plausible candidate for a cryptographic group supporting the eDDH assumption are groups based on elliptic curves. Despite over a decade or research, essentially no non-trivial attack is known on general elliptic curve groups. Therefore, the eDDH assumption on these groups appears to be a very reasonable assumption. We note that groups based on elliptic curves can be made public coin.

Construction. Our construction is as follows. We first describe the injective mode ($r = M$). Assume $M = 2^k$ is a power of two. Let $N = M^3 = 2^{3k}$.

- For $\lambda = 1, \ldots, k$, let $(\mathbb{G}_i, g_i, p_i) \leftarrow \texttt{GroupGen}(i)$.
- Let n_i be the smallest integer such that $p_i^{n_i} \geq N$.
- Choose pairwise independent uniform functions $h_i : \mathbb{G}_{i-1}^{k+n_{i-1}} \to \mathbb{Z}_{p_i}^{n_i}$ for $i = 2, \ldots, k$
- Choose random matrices $\mathbf{A}^{(i)} \in \mathbb{Z}_{p_i}^{(k+n_i)\times n_i}$. Let $\hat{\mathbf{A}}^{(i)} = g_i^{\mathbf{A}^{(i)}}$.
- Choose a pairwise independent uniform function h_1 from $[M]$ into $\mathbb{Z}_{p_1}^{n_1}$.
- Choose a pairwise independent uniform function $h_{i+1} : \mathbb{G}_i^{k+n_i} \to [N]$.
- Output $\{\mathbb{G}_i, g_i, p_i, h_i, \hat{\mathbf{A}}^{(i)}\}_{i\in[k]}$ as the description of f.

To evaluate f on input $x \in [M]$, do the following.

- Let $\mathbf{x}^{(1)} = h_1(x)$.
- For $i \in [k]$, let $\mathbf{y}^{(i)} \leftarrow \hat{\mathbf{A}}^{(i)} \cdot \mathbf{x}^{(i)} = g_i^{\mathbf{A}^{(i)}\cdot\mathbf{x}^{(i)}} \in \mathbb{G}_{p_i}^{k+n_i}$, $\mathbf{x}^{(i+1)} \leftarrow h_{i+1}(\mathbf{y}^{(i)})$.
- Output $\mathbf{x}^{(k+1)} \in [N]$

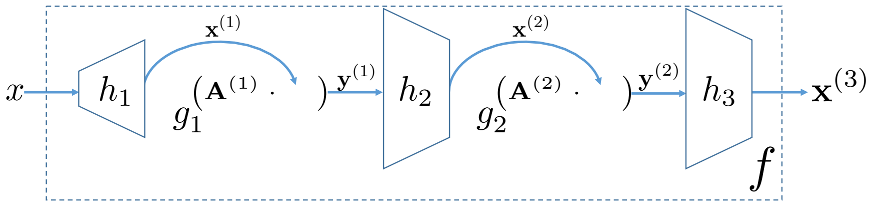

Figure 1: An example instantiation for $k = 2$.

A diagram of the evaluation of f is given in Figure 1. We observe that:

- For any f, the number of possible $\mathbf{y}^{(i)}$ values is always at most M. Therefore, since the h_i for $i > 1$ are chosen to be pairwise independent and random with co-domain of size at least $N = M^3$, the probability that there exists a collision in h_i is at most $M^2/N = 1/M$, which is negligible in $\log M$.
- With overwhelming probability, each of the $\mathbf{A}^{(i)}$ has full rank. Therefore, the operation $\mathbf{x}^{(i)} \to \hat{\mathbf{A}}^{(i)} \cdot \mathbf{x}^{(i)} = g_i^{\mathbf{A}^{(i)}\cdot\mathbf{x}^{(i)}}$ is injective.
- Thus, each step of the evaluation of f, and hence f itself, is injective, with overwhelming probability.
- The elements of $\hat{\mathbf{A}}^{(i)}$ are just random independent elements in the group. Therefore, if the group is public coin, these elements can be sampled in a public fashion. Therefore, the ELF is public coin.

We now describe the lossy modes of f. Given $r < M$, let i be the integer such that $2^i \in (r/4, r/2]$. The entire construction is the same, except that we choose $\mathbf{A}^{(i)}$ to be a random *rank 1* matrix. This means that the number of possible $\mathbf{A}^{(i)} \cdot \mathbf{x}^{(i)}$, and hence the number of possible $g_i^{\mathbf{A}^{(i)} \cdot \mathbf{x}^{(i)}}$, is at most $p_i < 2 \times 2^i \leq r$. Hence the image size is at most r, as desired.

For security, first we note that if an adversary can distinguish $\hat{\mathbf{A}}^{(i)} = g_i^{\mathbf{A}^{(i)}}$ for a random matrix $\mathbf{A}^{(i)}$ from the case of a random rank-1 matrix $\mathbf{A}^{(i)}$ with advantage ϵ, it is straightforward to obtain an adversary running in approximately the same time that distinguishes g_i, g_i^a, g_i^b, g_i^c from $g_i, g_i^a, g_i^b, g_i^{ab}$ with advantage $\epsilon/(n_i(k+n_i)) \geq \epsilon/12k^2$.

Let t be a polynomial in $k = \log M$, ϵ a non-negligible function in k. Let q be the polynomial guaranteed by the eDDH assumption. Let $q'(k) = 4q(t(k), \epsilon(k)/12k^2)$. Then for any $r \geq q'(k)$, we have that $i > q(t(k), \epsilon(k)/12k^2)$. This means no t-time adversary can distinguish g_i, g_i^a, g_i^b, g_i^c from $g_i, g_i^a, g_i^b, g_i^{ab}$ except with advantage at most $\epsilon/12k^2$. In turn, no t-time adversary can distinguish $\mathbf{A}^{(i)}$ being full rank or rank 1, except with advantage ϵ. Thus the lossy mode with image size r is indistinguishable from the injective mode, as desired.

Lastly, we describe an efficient image-enumeration procedure for our ELFs. As noted above, it suffices to devise an efficient algorithm that samples a random image point in the lossy mode. Our algorithm simply samples a random domain point, and then applies f to get the image point. To prove that this works, we make the following observations:

- Fix $\mathbf{A}^{(i)}$ to be rank-1, and let $S^{(i)}$ be the set of possible values of $\hat{\mathbf{A}}^{(i)} \cdot \mathbf{v} \in \mathbb{G}_{p_i}^{k+n_i}$. Note that $|S^{(i)}| = p_i$.

- Except for the step $\mathbf{y}^{(i)} = \hat{\mathbf{A}}^{(i)} \cdot \mathbf{x}^{(i)}$ for the i determined from r as above, all steps of the computation of f are injective. Thus, it suffices to show that, for a fixed f, on input a random x, $\mathbf{y}^{(i)}$ will be a random element in $S^{(i)}$.

- Fix the entire description of f except for h_i. Since all the prior steps are injective, and since $\mathbf{x}^{(i)}$ is the output of a pairwise independent function, the map $x \mapsto \mathbf{y}^{(i)}$ is pairwise independent. Let $\mathcal{D}_i$ be the output distribution of this map, which is the distribution of $\mathbf{y}^{(i)}$ under a random h_i for any fixed x. Then $\mathcal{D}_i$ has support p_i. Therefore, we can appeal to Lemma 2.2 and see that, given h, the distribution on $\mathbf{y}^{(i)}$ for a random x is within a distance $\frac{1}{2}\sqrt{\frac{1}{M}(p_i - 1)}$ of $\mathcal{D}_i$. This distance is negligible.

- It remains to show that the distribution $\mathcal{D}_i$ is random. Indeed, the uniformity of h_i guarantees that for any fixed x, $\mathbf{x}^{(i)}$ is uniform. Therefore, since the map $\mathbf{x}^{(i)} \mapsto \mathbf{y}^{(i)} = \hat{\mathbf{A}}^{(i)} \cdot \mathbf{x}^{(i)}$ is regular, $\mathbf{y}^{(i)}$ is uniform.

Thus we get the following theorem:

Theorem 3.7. *If there exists a cryptographic group where the eDDH assumption holds, then ELFs with efficiently enumerable images exist. Moreover, if the group is public coin, then so is the ELF.*

4 Point Function Obfuscation

A (expanding) random oracle H serves as a good point function obfuscator: to obfuscate the point function $I_x(x') = \begin{cases} 1 & \text{if } x' = x \\ 0 & \text{if } x' \neq x \end{cases}$, simply output $y = H(x)$. Then to run the "program" on input

x', simply check that $H(x') = y$. For any x that is drawn from an source with super-logarithmic min-entropy, an adversary making a polynomial number of queries to H will not be able to determine x from y. Thus, x is hidden to all efficient adversaries.

In this section, we show how to use ELFs to implement a concrete function H for which the strategy above still yields a secure point obfuscation (PO).

Definition 4.1. A point obfuscator (PO) is an efficient probabilistic algorithm $\mathcal{O}$ with the following properties:

- (Almost Perfect Correctness) On input a point function I_x, with overwhelming probability over the random coins of $\mathcal{O}$, $\mathcal{O}$ outputs the description of a program P that is functionally equivalent to I_x. P must run in time polynomial in the length of x and the security parameter.
- (Secrecy) For any distribution $\mathcal{D}$ over a set $\mathcal{X}$ with super-logarithmic min-entropy, the distribution $\mathcal{O}(I_x)$ for $x \leftarrow \mathcal{D}$ is computationally indistinguishable from $\mathcal{O}(I_{x'})$ where $x' \leftarrow U_{\mathcal{X}}$.

4.1 The Construction

Construction 4.2. Let $\mathcal{X}$ be the desired domain of H. To generate H, to the following:

- Let $\mathcal{Z}$ be some set such that $|\mathcal{X}|/|\mathcal{Z}|$ is negligible, and sample a hash function h from a uniform and pairwise independent function distribution from $\mathcal{X}$ to $\mathcal{Z}$. By setting $\mathcal{Z}$ even larger so that $|\mathcal{X}|^2/|\mathcal{Z}|$ is negligible, h will be injective with overwhelming probability for any pairwise independent function distribution. Alternatively, let $\mathcal{Z}$ be a large field and $|\mathcal{X}|$ a subset of $\mathcal{Z}$. Then we can set $h(x) = ax + b$ for random a, b. As long as $a \neq 0$, h is injective.
- Let $f \leftarrow \texttt{ELF.Gen}(|\mathcal{Z}|, |\mathcal{Z}|)$ to get an injective-mode f.
- Output $H = f \circ h$.

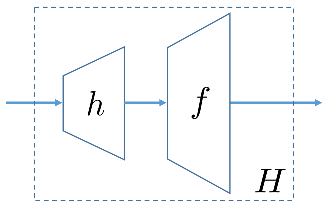

Figure 2: The function $H = f \circ h$.

Theorem 4.3. *Assuming* ELF *is a secure ELF, H in Construction 4.2 gives a secure point obfuscator. If* ELF *is public coin, then so is H.*

Proof. We will actually show something stronger: that the point function obfuscation of x is indistinguishable from an obfuscation of the all-zeros function. In particular, we will show that no efficient adversary can distinguish $y = f(h(x))$ from $y = f(z)$ for a uniformly random z. Notice that by injectivity of f, y has a pre-image under $H = f \circ h$ if and only if $z = f^{-1}(y)$ has a pre-image under h. Since we chose h to be expanding, when we sample z uniformly random, z will have no pre-image with overwhelming probability. Therefore, $y = f(z)$ has no pre-image with overwhelming probability.

The proof involves a sequence of hybrids. Suppose the adversary runs in time t and distinguishes $y = f(h(x))$ from $y = f(z)$ with non-negligible advantage ϵ.

Hybrid 0 This is the honestly generated $y = f(h(x))$ for f drawn in injective mode and x drawn from D.

Hybrid 1 Now, we change f to be lossy. That is, we generate $f \leftarrow \texttt{ELF.Gen}(|\mathcal{Z}|, r)$ where r is chosen so that no adversary running in time t can distinguish this lossy f from an injective f, except with advantage at most $\epsilon/3$. Thus by ELF security, the adversary cannot distinguish **Hybrid 0** from **Hybrid 1**, except with probability $\epsilon/3$.

Hybrid 2 Now we change y to be $y = f(z)$ for a random uniform $z \in \mathcal{Z}$. Fix f, and let E be the distribution of y. Then notice that by the pairwise independence and uniformity of h, the composition $H = f \circ h$ is pairwise independent and has output distribution E. Moreover, $\text{Supp}(E) \le r$ is a polynomial. Therefore, by using our new leftover hash lemma (Lemma 2.2), we see that **Hybrid 1** and **Hybrid 2** are indistinguishable, except with probability $\frac{1}{2}\sqrt{CP(D)(\ |\text{Supp}(E)| - 1\ )}$. As long as the collision probability of $\mathcal{X}$ is negligible (which in particular happens when $\mathcal{X}$ has superlogarithmic min-entropy), this quantity will be negligible. In particular, the distinguishing advantage will be less than $\epsilon/3$.

Hybrid 3 Now we change f to be injective again. Similarly to before, the distinguishing advantage between **Hybrid 2** and **Hybrid 3** will be at most $\epsilon/3$. Notice that **Hybrid 3** is exactly our all-zeros obfuscation. Therefore, **Hybrid 0** and **Hybrid 3** are indistinguishable, except with probability less than ϵ, contradicting our assumption about the adversary. □

In Section 6, we will show how to strengthen our construction to get a point obfuscator that is secure even against auxiliary information about the point.

5 Output Intractability

Consider any $k+1$-ary relation R over $\mathcal{Y}^k \times \mathcal{W}$ that is *computationally intractable*: on a random input $\mathbf{y} \in \mathcal{Y}^k$, it is computationally infeasible to find a $w \in \mathcal{W}$ such that $R(\mathbf{y}, w)$ outputs 1. If H is a random oracle, assuming k is a constant, it is computationally infeasible for find a set of distinct inputs $\mathbf{x}$, $x_i \neq x_j \forall i \neq j$, and a $w \in \mathcal{W}$, such that $R(H(\mathbf{x}), w) = 1$. We will now show how to build standard-model hash functions H that achieve the same property.

Definition 5.1. A family of hash functions $H : \mathcal{X} \to \mathcal{Y}$ is *k-ary output intractable* if, for any computationally intractable $k+1$-ary relation $R : \mathcal{Y}^k \times \mathcal{W} \to \{0,1\}$, no efficient adversary, given H, can find a set of distinct inputs $\mathbf{x} \in \mathcal{X}^k$ and an element $z \in \mathcal{W}$, such that $R(H(\mathbf{x}), w) = 1$.

Note that binary output intractability implies as a special case collision resistance. In the unary case, and if $\mathcal{Z}$ is just a singleton set, then output intractability is a special case of *correlation intractability*, where the relation in addition depends on the *input*.

The unary case captures the following use case of hash functions: a given protocol may require a common reference string (crs), but some or all instances of the crs may admit a trapdoor that allows breaking the protocol. Of course, such a trapdoor should be difficult to find for a random crs. To "prove" that the crs is generated so that the generator of the crs does not know a trapdoor, the generator sets the crs to be the output of a public hash function on an arbitrary point. Since the potentially malicious generator does not control the hash function, he should be unable to find an output along with a corresponding trapdoor. Modeling the hash function as a random oracle, this methodology is sound. However, standard notions of security do not prevent the crs generator from choosing the input in such a way so that it knows a trapdoor. Unary output intractability precludes this case. Of course, the hash function itself needs to be set up in a trusted manner; however, once the hash function is set up and trusted, it can be used to generate arbitrarily many different crs by even untrusted authorities.

5.1 The Construction

Construction 5.2. Let $\mathcal{X}$ be the desired domain of H, and $\mathcal{Y}$ the desired range. To generate H, to the following:

- Let $f \leftarrow \texttt{ELF.Gen}(|\mathcal{X}|, |\mathcal{X}|)$ to get an injective-mode f, with codomain $\mathcal{Z}$.
- Let g be a k-wise independent and uniform function from $\mathcal{Z}$ to $\mathcal{Y}$.
- Output $H = g \circ f$.

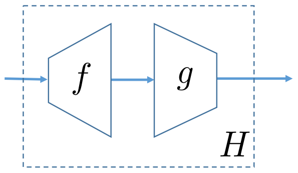

Figure 3: The function $H = g \circ f$.

Theorem 5.3. *If* ELF *is a secure ELF with an efficiently enumerable image, then for any constant k the hash function H in Construction 5.2 is k-ary output intractable. If* ELF *is public coin, then so is H.*

Proof. Suppose toward contradiction that there is an intractable $k+1$-ary relation R and an adversary $\mathcal{A}$ that on input H finds a set of distinct inputs $\mathbf{x}$ and a value $w \in \mathcal{W}$ such that $R(H(\mathbf{x}), w) = 1$ with non-negligible probability ϵ. We will switch to a lossy mode for f so that (1) f has polynomial image size, and (2) no adversary running in time t (for a t to be chosen later) can distinguish the lossy mode from injective, except with probability $\epsilon/3$. By choosing t to be larger than the running time of $\mathcal{A}$, we have that $\mathcal{A}$ still outputs $\mathbf{x}$ of distinct elements, and a string w, such that $R(H(\mathbf{x}), w) = 1$ with non-negligible probability $2\epsilon/3$.

We first argue that each of the elements of $f(\mathbf{x})$ are distinct except with probability $\epsilon/3$. Since this was true in the injective case (since $\mathbf{x}$ is distinct), if this is not true in the lossy case, then the injective and lossy modes could be easily distinguished by an adversary taking slightly more time than $\mathcal{A}$. Let t be this time, so that this distinguisher is impossible. Thus, the adversary succeeds *and* the elements of $f(\mathbf{x})$ are distinct with probability at least $\epsilon/3$. Let S be the polynomial-sized set of image points of f. Then in other words, the adversary comes up with an ordered set $\mathbf{z}$ of distinct elements in S, and a string w, such that $R(g(\mathbf{z}), w) = 1$.

Now, note that, for any ordered set $\mathbf{z}$ of k distinct inputs, $g(\mathbf{z})$ is distributed uniformly at random, by the k-wise independence of g. Moreover, it is straightforward, given $\mathbf{z}$ and a vector $\mathbf{y} \in \mathcal{Y}^k$, to sample a random g conditioned on $g(\mathbf{z}) = \mathbf{y}$. Sampling random $\mathbf{y}$, and then g in this way, gives a correctly distributed g.

We now describe an algorithm $\mathcal{B}$ that breaks the intractability of R. $\mathcal{B}$, on input $\mathbf{y} \in \mathcal{Y}^k$, chooses lossy f as above, and then selects a random ordered set $\mathbf{z}, |\mathbf{z}| = k$ among the p outputs of f. Next, it chooses a random g such that $g(\mathbf{z}) = \mathbf{y}$. Finally, it runs $\mathcal{A}$ on the hash function $H = g \circ f$. When $\mathcal{A}$ outputs $\mathbf{x}, w$, if $f(\mathbf{x}) = \mathbf{z}$ (equivalently, $H(\mathbf{x}) = \mathbf{y}$), $\mathcal{B}$ outputs w; otherwise it aborts.

Since $\mathbf{y}$ is hidden from $\mathcal{A}$'s view, g is distributed randomly according to the k-wise independent distribution. Therefore, $\mathcal{A}$ will output a valid w with probability at least $\epsilon/3$. If $\mathcal{B}$'s guess for $\mathbf{z}$ was correct, then w will break the correlation intractability of R on $\mathbf{y}$. Since $\mathbf{z}$ is random and independent of $\mathcal{A}$'s view, the probability of a good guess is at least $1/p^k$ (since there are at most p^k such $\mathbf{z}$). Therefore, $\mathcal{B}$ breaks the intractability of R with probablity $\epsilon/3p^k$, which is non-negligible. $\square$

6 Leakage-resilient PRGs, AIPO and Poly-many Hardcore Bits

In this section, we use ELFs to give arbitrarily-many hardcore bits for any one-way function, or for constructing point function obfuscation secure in the presence of auxiliary information. Both of these can be seen as special cases of a very strong security requirement for pseudorandom generators.

Definition 6.1. A distribution $\mathcal{D}$ on pairs $(x, z) \in \mathcal{X} \times \mathcal{Z}$ is *computationally unpredictable* if no efficient adversary can guess x given z.

Definition 6.2. A family of pseudorandom generators $H : \mathcal{X} \to \mathcal{Y}$ secure for computationally unpredictable seeds if, for any computationally unpredictable distribution on $(\mathcal{X}, \mathcal{Z})$, no efficient adversary can distinguish $(H, z, H(x))$ from (H, z, S) where $(x, z) \leftarrow \mathcal{D}$ and $S \leftarrow U_{\mathcal{Y}}$.

Basically, this requirement states that H is a secure pseudorandom generator for arbitrary distributions on the seed, and even remains secure in the presence of arbitrary leakage about the seed, so long as the seed remains *computationally* unpredictable. The only restriction is that the distribution on the seed and the leakage must be chosen independently of H. However, in the absence of other restrictions, this independence between the source $\mathcal{D}$ and function H can easily be seen to be necessary: if z contained a few bits of $H(x)$, then it is trivial to distinguish $H(x)$ from random.

6.1 The Construction

The intuition behind our construction is the following. The usual way of extracting pseudorandomness from computationally unpredictable source is to output a hardcore bit of the source, say using Goldreich-Levin [GL89]. While this can be used to generate a logarithmic number of pseudorandom bits, security is lost once a super-logarithmic number of hardcore bits have been generated in this way.

In order to get around this logarithmic barrier, we actually compute a *polynomial* number of Goldriech-Levin bits. Of course, we cannot output these in the clear or else the seed can be easily computed by linear algebra. Instead, we scramble the hardcore bits using a sequence of ELFs. We can argue that each of the (scrambled) hardcore bits really is "as good as" random, in the sense that we can replace each bit with a truly random bit before scrambling without detection. To do so, we use the lossiness of the ELFs to argue that, when the ith hardcore bit is incorporated into the scramble, enough information is lost about the previous bits that the ith bit actually still is hardcore. By iterating this for each bit, we replace each one with random. We now give the details.

Construction 6.3. Let q be the input length and m be the output length. We will consider inputs x as q-dimensional vectors $\mathbf{x} \in \mathbb{F}_2^q$. Let $\texttt{ELF}$ be an ELF. Let $M = 2^{m+1}$, and let n be the bit-length of the ELF on input $m+1$. Set $N = 2^n$. Let ℓ be some polynomial in m to be determined later. First, we will construct a function H' as follows.

Choose random $f_1, \dots, f_\ell \leftarrow \texttt{ELF.Gen}(M, M)$ where $f_i : [M] \to [N]$, and let $h_1, \dots, h_\ell : [N] \to [M/2] = [2^m]$ be sampled from a pairwise independent and uniform function family. Define $\mathbf{f} = \{f_1, \dots, f_\ell\}$ and $\mathbf{h} = \{h_1, \dots, h_\ell\}$. Define $H_i' : \{0,1\}^i \to [N]$ as follows:

- $H_0'() = 1 \in [2^m]$
- $H_i'(\mathbf{b}_{[1,i-1]}, b_i)$: compute $y_i = H_{i-1}'(\mathbf{b}_{[1,i-1]})$, $z_i \leftarrow f_i(y_i || b_i)$, and output $y_{i+1} \leftarrow h_i(z_i)$

Then we set $H' = H_\ell'$. Then to define H, choose a random matrix $\mathbf{R} \in \mathbb{F}_2^{\ell \times q}$. The description of H consists of $\mathbf{f}, \mathbf{h}, \mathbf{R}$. Then set $H(x) = H'(\mathbf{R} \cdot \mathbf{x})$. A diagram of H is given in Figure 4.

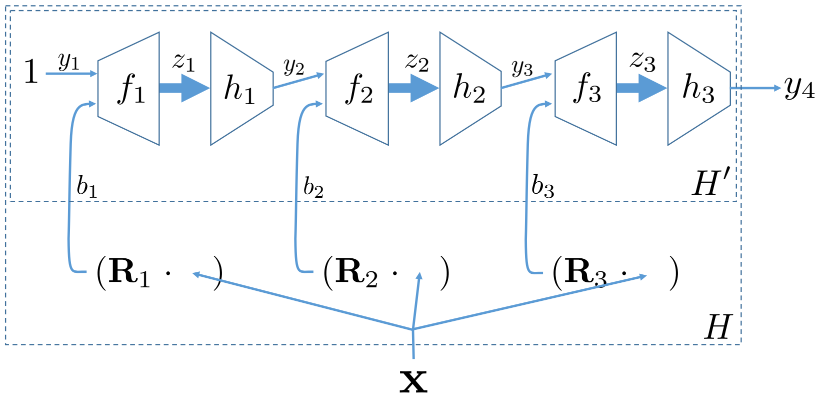

Figure 4: An example instantiation for $\ell = 3$.

We now prove two important facts about H and H':

Claim 6.4. *If $\ell \geq m$, $H'(\mathbf{b})$ is statistically close to uniform for uniform* $\mathbf{b}$.

Proof. Let c_i be the collision probability of z_i and d_i the collision probability for y_{i+1} when the b_i are chosen at random. Then $c_i = d_{i-1}/2$, $d_0 = 1$, and $d_i \leq c_i + 1/2^n$. Solving the recurrence, we see that $d_i \leq \frac{2^m + 2^{i+1} - 2}{2^{m+i}}$. Setting $i = \ell = n$, we see that the collision probability of y_{m+1}, namely d_m, is at most 3×2^{-m}. Now, recall that the collision probability of a distribution is simply the Rényi divergence between the distribution and uniform. Then, applying Lemma 2.1, we see that the distribution of y_{m+1} is within a negligible distance $\sqrt{3} \times 2^{-m/2-1}$ from uniform. □

We will thus set $\ell = m$ in our construction of H'. Claim 6.4 will be crucial for our proof that H meets our strong PRG security notion.

We also show our H is injective (whp) exactly when a truly random function with the same domain and co-domain is injective (whp).

Claim 6.5. *If $2^{-(m-2q)}$ is negligible (in q), and $\ell \geq m$, then with overwhelming probability H is injective.*

Proof. First, note that with overwhelming probability by our choice of $\ell \geq m \geq 2q$, $\mathbf{R}$ is full rank. Next, let $\mathcal{Y}_i$ be the set of possible y_i values as we vary $\mathbf{x}$, and $\mathcal{Z}_i$ be the set of possible z_i values. By the injectivity of f_i, we have that $|\mathcal{Z}_i| \geq |\mathcal{Y}_i|$. Moreover, since h_i is pairwise independent and uniform, with overwhelming probability h_i is injective on $\mathcal{Z}_i$ since $|\mathcal{Z}_i| \leq 2^q$ but the co-domain of h_i has size $2^m \gg (2^q)^2$. Therefore $|\mathcal{Y}_{i+1}| = |\mathcal{Z}_i| \geq |\mathcal{Y}_i|$. This means that as we increase i, the image size never decreases.

Now pick q linearly independent rows of $\mathbf{R}$. We will assume that the q rows constitute the first q rows of $\mathbf{R}$; the more general case is handled analogously. By performing an appropriate invertible transformation on the domain, we can assume that these q rows form the identity matrix. Therefore, we can take $b_i = x_i$ for $i \in [q]$. Next, observe that y_i for $i \in [q]$ only depends on the first $i-1$ bits of $\mathbf{x}$. Thus the set of possible pairs $(y_i, b_i) = (y_i, x_i)$ is exactly $\mathcal{Y}_i \times \{0,1\}$, which has size $2|\mathcal{Y}_i|$. By the injectivity of f_i, $|\mathcal{Z}_i| = 2|\mathcal{Y}_i|$. Since $|\mathcal{Y}_{i+1}| = |\mathcal{Z}_i| = 2|\mathcal{Y}_i|$, we have that the image size exactly doubles in each iteration for $i \in [q]$. Once we get to $i = q$, the image size is 2^q, and the remaining iterations do not introduce any collisions. Thus the image size of H is 2^q, meaning H is injective. $\square$

We now present our main theorem of the section:

Theorem 6.6. *If `ELF` is a secure ELF, then H in Construction 6.3 is a pseudorandom generator secure for computationally unpredictable seeds. If `ELF` is public coin, then so is H.*

Proof. Recall that $H(\mathbf{x}) = H'(\mathbf{R} \cdot \mathbf{x})$, and that $H'(\mathbf{b})$ is statistically close to random when $\mathbf{b}$ is random. Therefore, it suffices to show that the following distributions are indistinguishable: $(\ \mathbf{f}, \mathbf{h}, \mathbf{R}, z, H'(\mathbf{R} \cdot \mathbf{x})\ )$ and $(\ \mathbf{f}, \mathbf{h}, \mathbf{R}, z, H'(\mathbf{b})\ )$ for a uniformly random $\mathbf{b}$.

Suppose an adversary $\mathcal{A}$ has non-negligible advantage ϵ in distinguishing the two distributions. Define $\mathbf{b}^{(i)}$ so that the first i bits of $\mathbf{b}^{(i)}$ are equal to the first i bits of $\mathbf{R} \cdot \mathbf{x}$, and the remaining $\ell - i$ bits are chosen uniformly at random independently of $\mathbf{x}$. Define **Hybrid** i to be the case where $\mathcal{A}$ is given the distribution $(\ \mathbf{f}, \mathbf{h}, \mathbf{R}, z, H'(\mathbf{b}^{(i)})\ )$.

Then $\mathcal{A}$ distinguishes **Hybrid 0** from **Hybrid** ℓ with probability ϵ. Thus there is an index $i \in [\ell]$ such that the adversary distinguishes **Hybrid** $i-1$ from **Hybird** i with probability at least ϵ/ℓ. Next, observe that since bits $i+1$ through t are random in either case, they can be simulated independently of the challenge. Moreover, $H'(\mathbf{b})$ can be computed given $H'_{i-1}(\mathbf{b}_{[i-1]})$, b_i (be it random or equal to $\mathbf{R}_i, \mathbf{x}$), and the random $b_{i+1}, \dots, b_\ell$. Thus, we can construct an adversary $\mathcal{A}'$ that distinguishes the following distributions:

$$(\ \mathbf{f}, \mathbf{h}, \mathbf{R}_{[i-1]}, z, H'_{i-1}(\mathbf{R}_{[i-1]} \cdot \mathbf{x}), \mathbf{R}_i, \mathbf{R}_i \cdot \mathbf{x}\ ) \text{ and } (\ \mathbf{f}, \mathbf{h}, \mathbf{R}_{[i-1]}, z, H'_{i-1}(\mathbf{R}_{[i-1]} \cdot \mathbf{x}), \mathbf{R}_i, b_i\ )$$

with advantage ϵ/t, where $\mathbf{R}_{[i-1]}$ consists of the first $i-1$ rows of $\mathbf{R}$, $\mathbf{R}_i$ is the ith row of $\mathbf{R}$, and b_i is a random bit.

Next, let $\epsilon' = \epsilon/3t$, which is a polynomial. There is a polynomial r such $\mathcal{A}'$ cannot distinguish f_i generated as $\texttt{ELF.Gen}(M, r)$ from the honest f_i generated from $\texttt{ELF.Gen}(M, M)$, except with probability at most ϵ'. This means, if we generate $f_i \leftarrow \texttt{ELF.Gen}(M, r)$, we have that $\mathcal{A}'$ still distinguishes the distributions

$$(\ \mathbf{f}, \mathbf{h}, \mathbf{R}_{[i-1]}, z, H'_{i-1}(\mathbf{R}_{[i-1]} \cdot \mathbf{x}), \mathbf{R}_i, \mathbf{R}_i \cdot \mathbf{x}\ ) \text{ and } (\ \mathbf{f}, \mathbf{h}, \mathbf{R}_{[i-1]}, z, H'_{i-1}(\mathbf{R}_{[i-1]} \cdot \mathbf{x}), \mathbf{R}_i, b_i\ )$$

with advantage ϵ'. Put another way, given $(\ \mathbf{f}, \mathbf{h}, \mathbf{R}_{[i-1]}, z, H'_{i-1}(\mathbf{R}_{[i-1]} \cdot \mathbf{x}), \mathbf{R}_i\ )$, $\mathcal{A}'$ is able to compute $\mathbf{R}_i \cdot \mathbf{x}$ with probability $\frac{1}{2} + \epsilon'$.

Now fix $\mathbf{f}, \mathbf{h}, \mathbf{R}_{[i-1]}$, which fixes H'_{i-1}. Let $y_i = H'_{i-1}(\mathbf{R}_{[i-1]} \cdot \mathbf{x})$. Notice that since $\mathbf{f}, \mathbf{h}$ are fixed, there are at most r possible values for y_i, and recall that r is a polynomial. We now make the following claim:

Claim 6.7. *Let $\mathcal{D}$ be a computationally unpredictable distribution on $\mathcal{X} \times \mathcal{Z}$. Suppose $T : \mathcal{X} \to \mathcal{R}$ is drawn from a family $\mathcal{T}$ of efficient functions where the size of the image of T is polynomial. Then the following distribution is also computationally unpredictable: $(\ x,\ (T, z, T(x))\ )$ where $T \leftarrow \mathcal{T}$, $(x, z) \leftarrow \mathcal{D}$.*

Proof. Suppose we have an efficient adversary $\mathcal{B}$ that predicts x with non-negligible probability δ given $T, z, T(x)$, and suppose T has polynomial image size r. We then construct a new adversary $\mathcal{C}$ that, given x, samples a random T, samples $(x', z') \leftarrow \mathcal{D}$, and sets $a = T(x')$. It then runs $\mathcal{B}(T, z, a)$ to get a string x'', which it outputs. Notice that a is sampled from the same distribution as $T(x)$, so with probability at least $1/r$, $a = T(x)$. In this case, $x'' = x$ with probability δ. Therefore, $\mathcal{C}$ outputs x with probability δ/r, which is non-negligible. □

Using Claim 6.7 with $T = H'_{i-1}(\mathbf{R}_{[i-1]} \cdot \mathbf{x})$, we see that $(x,\ (\mathbf{f}, \mathbf{h}, \mathbf{R}_{[i-1]}, z, H'_{i-1}(\mathbf{R}_{[i-1]} \cdot \mathbf{x}))\ )$ is computationally unpredictable. Moreover, $\mathbf{R}_i \cdot \mathbf{x}$ is a Goldriech-Levin [GL89] hardcore bit for any computationally unpredictable source. Hence, no efficient adversary can predict $\mathbf{R}_x \cdot \mathbf{x}$ given $(\mathbf{f}, \mathbf{h}, \mathbf{R}_{[i-1]}, z, H'_{i-1}(\mathbf{R}_{[i-1]} \cdot \mathbf{x}), \mathbf{R}_i)$, except with negligible probability. This contradicts the existence of $\mathcal{A}'$, proving Theorem 6.6. □

6.2 Applications

Polynomially-many hardcore bits for any one-way function. We see that H immediately gives us a hardcore function of arbitrary stretch for any computationally unpredictable distribution. This includes any one-way function. To the best of our knowledge, this is the first hardcore function of arbitrary stretch for general computationally unpredictable sources. In the special case of one-way functions, the only prior constructions are due to Bellare, Stepanovs, and Tessaro [BST14] using differing inputs obfuscation (diO), and of Zhandry [Zha16] using extractable witness PRFs. Both diO and extractable witness PRFs constitute very strong knowledge assumptions, and the plausibility of both have been significantly challenged [GGHW14]. In the case of *injective* one-way functions, [BST14] can relax the assumption to indistinguishability obfuscation (iO), which still remains a very strong primitive based on very new and relatively untested assumptions on multilinear maps. Our construction offers an entirely different approach to constructing hardcore functions with arbitrary stretch, and is based on a very simple primitive. While the only instantiation of our primitive so far requires less-than-standard elliptic curve assumptions, elliptic curves and assumptions on them, including ours, are far better understood.

Strong injective one-way functions. Bitansky and Paneth [BP12] conjecture the existence of a very strong one-way permutation family. We demonstrate that our function H meets this notion of security. Unfortunately, however, it is only injective, not a permutation.

Definition 6.8. A [BP12] permutation is a family of functions H such that for any computationally unpredictable distribution $\mathcal{D}$, the following two distrubitons are also unpredictable:

$$(x,\ (z, H, H(x))\ ), \text{ and } (H(x),\ (z, H)\ ) \quad (\text{where } (x, z) \leftarrow \mathcal{D})$$

The first property is a generalization of a strong uninvertability assumption of Wee [Wee05]. The second guarantees that if x is unpredictable, then so is $H(x)$. We now show that our construction H satisfies this definition:

Theorem 6.9. *H constructed above using a secure ELF, when set to be injective as in Claim 6.5, is a [BP12] injective one-way function.*

Proof. For the first property, suppose that an efficient adversary $\mathcal{A}$ can predict x given $(z, H, H(x))$. We now build an adversary $\mathcal{A}'$ that can distinguish $(z, H, H(x))$ from (z, H, S) for a random S. $\mathcal{A}'(z, H, S)$ runs $\mathcal{A}(z, H, S)$ to get a point x'. It then tests if $H(x') = S$. If so, it outputs 1 and otherwise outputs 0. In the case that $S = H(x)$, by assumption $\mathcal{A}$ will output $x' = x$ with non-negligible probability, meaning $\mathcal{A}'$ will output 1 with non-negligible probability. In the case that S is random, since H is expanding, with overwhelming probability there is no pre-image of S under H. Therefore $\mathcal{A}$ will output 1 with negligible probability. Thus $\mathcal{A}'$ successfully distinguishes the two cases, violating Theorem 6.6.

For the second property, suppose an adversary $\mathcal{A}$ can predict $H(x)$ from z, H. We construct an adversary $\mathcal{A}'$ that can distinguish $(z, H, H(x))$ from (z, H, S) for a random S. $\mathcal{A}'(z, H, S)$ runs $\mathcal{A}(z, H)$ to get a string S', and then outputs 1 if and only if $S' = S$. In the case where $S = H(x)$, by assumption $\mathcal{A}$ will output S, and so $\mathcal{A}'$ will output 1, with non-negligible probability. In constrast, when S is chosen at random it is independent of the view of $\mathcal{A}$. Therefore $\mathcal{A}$ will output S, and so $\mathcal{A}'$ will output 1, with negligible probability. Thus $\mathcal{A}'$ successfully distinguishes the two cases, violating Theorem 6.6. □

The main application of Bitansky and Paneths [BP12] assumption is to build auxiliary input point function obfuscation (AIPO). Since H is not a permutation, it cannot be immediately plugged into their construction. Yet, next, we show that going through their construction is unnecessary in our case: we show that our function H gives an AIPO "out of the box" with no additional overhead.

Point function obfuscation with auxiliary input (AIPO). We now show how to achieve full AIPO using just the assumption of ELFs.

Definition 6.10. A *auxiliary input point obfuscator* (AIPO) is an efficient probabilistic algorithm $\mathcal{O}$ that saitsfies the *almost perfect correctness* requirement of Definition 4.1, as well as the following secrecy requirement: for any unpredictable distribution $\mathcal{D}$ over pairs $(x, z) \in \mathcal{X} \times \mathcal{Z}$, the following distributions are computationally indistinguishable:

$$(\mathcal{O}(I_x), z) : (x, z) \leftarrow \mathcal{D} \text{ and } (\mathcal{O}(I_{x'}), z) : (x, z) \leftarrow \mathcal{D}; x' \leftarrow \mathcal{X}$$

As in Section 4, an expanding ideal hash function (random oracle) H gives a very natural AIPO: the obfuscation of a point function I_x is simply $S = H(x)$. Injectivity of H gives (almost perfect) correctness. Moreover, security is easily proved in the random oracle model.

We now show that that by choosing H to be as in the construction above, the same is true. In particular, by Claim 6.5, H is injective in the same regime of input/output sizes as a random oracle. For security, we have the following:

Theorem 6.11. *The obfuscation construction described above is a secure AIPO assuming H is constructed as in Construction 6.3 using a secure ELF.*

Proof. Note that since H is expanding, if we choose S at random from $[2^m]$, then with overwhelming probability there are no inputs $\mathbf{x}$ that map to S. Therefore, the obfuscated program corresponding to S is just the all-zeros function.

Let $\mathcal{D}$ be any computationally unpredictable source. We thus need to show that the following two distributions are indistinguishable: $(\ H, z, H(\mathbf{x})\ )$ and$(\ H, z, S\ )$ (where $(\mathbf{x}, z) \leftarrow \mathcal{D}$). This follows immediately from Theorem 6.6. □

Public key encryption from trapdoor permutations. We show how our hardcore function can be used in the hybrid encryption schemes of Bellare and Rogaway [BR93] for converting a trapdoor permutation into a public key encryption scheme. Recall the [BR93] public key encryption scheme:

- $\texttt{PKE.Gen}(\lambda)$: run $(P, P^{-1}) \leftarrow \texttt{TDP.Gen}(\lambda)$. Choose a random H. Then the secret key for the scheme is $\texttt{sk} = (P^{-1}, H)$ and the public key is $\texttt{pk} = (P, H)$.
- $\texttt{PKE.Enc}(\texttt{pk}, m)$: Choose a random r in the domain of P. Output $c = (P(r), H(r) \oplus m)$. That is, the ciphertext has two components, $P(r)$, and a one-time pad encryption of m using $H(r)$ as the secret key.
- $\texttt{PKE.Dec}(\texttt{sk}, c = (c_1, c_2))$: let $r = P^{-1}(c_1)$, and then output $m = H(r) \oplus c_2$.

The proof of the following trivially follows from Theorem 6.6 and the fact that P is a one-way function so that $H(r)$ is pseudorandom given $P(r)$:

Theorem 6.12. *If P is a secure trapdoor permutation and H is constructed as in Construction 6.3 using a secure ELF, then* PKE *described above is a secure public key encryption scheme.*

7 Full Domain Hash Signatures

The Full Domain Hash signature scheme [BR93] makes use of a trapdoor permutation P with trapdoor P^{-1}, and a random oracle O whose range is the same as the domain of P. The signature on a message m is the pre-image under P of $H(m)$: $P^{-1}(H(m))$. Verifying the signature is straightforward. The proof of security in the random oracle model works roughly as follows. We are given a challenge $y = P(x)$ for some unknown x, and our goal is to use a forger to find x. We need to embed our challenge y into the forger's view. We do this by programming the random oracle seen by the adversary: at a randomly chosen oracle query (say on message m^*), we set the output of H to be y (the rest of the outputs are sampled at random). If the adversary makes q queries, with probability $1/q$, the adversary's forgery will be on m^*. In this case, the forgery is the pre-image of y, namely x. Thus, with probability $1/q$ the forger enables us to invert the trapdoor perpetuation.

Unfortunately, this proof technique seems to inherently require programming the random oracle H, and the programming needs to be *adaptive* (since we do not know a priori which points the adversary will query on). There is no standard model analog for such programming: for any concrete implementation of H, once we publish H (which must be a part of the public key), we have fixed the value of H at all messages.

One alternative is to settle for *selective* security for the signature scheme, where the adversary commits to the message m he will sign before receiving the public key. Here, H only needs to be program at m, which will be known when generating H. Therefore, a standard-model instantiation may be possible. Indeed, Hohenberger, Sahai, and Waters [HSW14] show that this is possible assuming indistinguishability obfuscation. Adaptive security can be obtained at the cost of an exponential loss in the security reduction, therefore requiring the exponential hardness of *all* primitives involved, including the trapdoor permutation and the indistinguishability obfuscator. [HSW14] additionally show how to obtain adaptive security in the particular case where the trapdoor permutation is instantiated with the RSA permutation.

Beyond achieving only static security for general TDPs, another limitation of [HSW14] is that the hash function code depends not only on the TDP, but also on the specific public key for the TDP that the hash function will be used with. Therefore, H cannot be a global reusable hash function, but must instead be re-instantiated for every separate public key. This is in contrast to typical heuristic instantiations of the random oracle used in practice, which use public standardized hash functions such as SHA256.

We note that again, complexity leveraging provides a solution to this limitation as well. The idea is to start with a universal H, and then in the reduction convert it to a permutation-dependent H as in [HSW14], and proceed using their proof to argue security. The reduction involves stepping through every input to H one at a time to change the output on that particular input to match [HSW14]; hence the exponential loss.

However, using complexity leveraging in this setting, besides having to assume that the various primitives involved are *very* secure, has a practical implication. If some trusted authority were to generate a global hash function as above, it likely will not be useable with existing trapdoor permutation keys, at least not in a sound manner. The issue is that, due to the exponential loss, all the building blocks, including the trapdoor permutation, need to have key sizes set appropriately large to as to handle this loss. Most currently deployed cryptographic keys are *not* set with this loss in mind. Therefore, in order to use this new hash function, all users would have to republish new, longer keys, which is a significant administrative hurdle.

In contrast, if the permutation was only required to be polynomially secure, existing deployed keys for trapdoor permutations could be used with the new hash function.

To avoid the exponential loss in the reduction, we now describe another proof strategy, derived from Zhandry [Zha12], that does not require adaptive programming, and is independent of the TDP and public key (except for depending on a size bound for the TDP evaluation circuit). Suppose the adversary makes q_H oracle queries to H. Consider the following non-uniform distribution on H: first sample $r = O(q_H^{1/2})$ random values $y_1, \dots, y_r$. Then for each input x, set the output $H(x)$ to be chosen randomly from $\{y_i\}_{i\in[r]}$. Put another way, define H to be the composition $H = H_2 \circ H_1$, where $H_1 : \mathcal{X} \to [r]$ and $H_2 : [r] \to \mathcal{Y}$. Following Zhandry, we call functions from this distribution *small-range* functions.

From the adversary's perspective, as long at none of the q queries form a collision in H, such small range functions are indistinguishable from the truly random case. Moreover by our choice of r, with probability $1/2$, there will be no collisions in the small-range case. Thus, if our adversary forges with probability 1 in the truly random case, he will still forge with probability $1/2$ in the small-range case.

Now, rather than choosing one of the adversary's queries at random to insert y, we instead set one of the y_i to by y. That is, we choose a random $i \in [r]$, and set $y_i = y$. For all other i, we set $y_i = P(x_i)$ for randomly chosen x_i. As long as the adversary never makes a signature query on an message m such that $H(m) = y$, the reduction will be able to produce the signature as one of the x_i. Let q_S be the number of signing queries. We can assume that for every signing query on message m, the adversary also makes a oracle query to H on m, so that $q_S \leq q_H$. Then the probability that we can answer all signing queries is at least $1 - (1 - 1/q_H)^{q_S} \geq 1 - (1 - 1/q_S)^{q_S} \geq 1 - 1/e$. Moreover, with probability $1/r$, a forgery produced by the adversary will invert y. Therefore, with non-negligible probability, we can successfully use the adversary to invert the permutation.

Boneh and Zhandry [BZ13] uses this flavor of proof strategies to prove the security of several signature schemes against certain models of quantum attacks. Even though we are not concerned with quantum attacks, the proof strategy is still useful in our context because now the random oracle can be programmed *statically* — all of the outputs of O can be chosen up front. Now we may hope for a standard-model instantiation of O that allows for this proof technique to work. Indeed, by setting H_1 to be an ELF (thus getting the polynomial image size), and H_2 to be an appropriate function, we can emulate this proof strategy.

7.1 Definitions

Puncturable PRFs. An $(\mathcal{X}, \mathcal{Y})$-pseudorandom function `PRF` with domain $\mathcal{X}$ and co-domain $\mathcal{Y}$ consists of a polynomial-time algorithm `PRF.Gen`() that outputs the description of an efficiently computable function $g : \mathcal{X} \to \mathcal{Y}$. The usual security requirement for a PRF is that oracle access to a random $g \leftarrow$ `PRF.Gen`() is indistinguishable from oracle access to a truly random function from $\mathcal{X}$ to $\mathcal{Y}$.

A puncturable PRF has an additional algorithm `PRF.Punct` that takes as input g and a domain point $x \in \mathcal{X}$, and outputs a "punctured key", namely the description of a function $g^x : \mathcal{X} \to \mathcal{Y} \cup \{\perp\}$ where $g^x(y) = \begin{cases} g(y) & \text{if } y \neq x \\ \perp & \text{if } y = x \end{cases}$. Punctured PRF security is the following: the adversary chooses a point $x \in \mathcal{X}$, and receives g^x where $g \leftarrow$ `PRF.Gen`(). Then the adversary, given g^x, is tasked with distinguishing $g(x)$ from a random string in $\mathcal{Y}$.

Indistinguishability Obfuscation. An *indistinguiability obfuscator* `IO` is a PPT uniform algorithm satisfying the following conditions:

- `IO`(C) preserves the functionality of C. That is, for any polynomial-sized circuit C, if we compute $C' =$ `IO`(C), then $C'(x) = C(x)$ for all inputs x.
- For any two circuits C_0, C_1 with the same functionality and size, the circuits `IO`(C_0) and `IO`(C_1) are computationally indistinguishable.

The first candidate construction of such obfuscators is due to Garg et al. [GGH+13].

7.2 The Construction

Construction 7.1. Let $\mathcal{M}$ be the domain of H (the message space), and $\mathcal{Y}$ be the co-domain. To generate H, to the following:

- Let $\texttt{ELF}$ be an ELF. Let $f \leftarrow \texttt{ELF.Gen}(|\mathcal{M}|, |\mathcal{M}|)$ to get an injective-mode f.
- Let $\texttt{PRF}$ be a puncturable PRF with domain $\mathcal{Z}$ and co-domain $\mathcal{X}$. Let $g \leftarrow \texttt{PRF.Gen}()$.
- Let $\texttt{IO}$ be an indistinguishability obfuscator.
- Output $H = \texttt{IO}(g \circ f)$, where $g \circ f$ is appropriately padded before obfuscating.

Security Proof For FDH Signatures. We recall the full-domain hash (FDH) signature scheme using a hash function H. The starting point for the scheme is trapdoor permutation $\texttt{TDP}$, which is used to build a signature scheme $\texttt{SIG}$.

- $\texttt{SIG.Gen}()$: Run $(P, P^{-1}) \leftarrow \texttt{TDP.Gen}()$. Output $\texttt{sk} = P^{-1}$ and $\texttt{pk} = P$.
- $\texttt{SIG.Sign}(\texttt{sk}, m)$: run $\tilde{m} \leftarrow H(m)$ and output $\sigma \leftarrow P^{-1}(\tilde{m})$
- $\texttt{SIG.Ver}(\texttt{pk}, m, \sigma)$: run $\tilde{m} \leftarrow H(m)$, and verifies that $P(\sigma) = \tilde{m}$.

Theorem 7.2. *Assuming that* $\texttt{ELF}$ *is a secure ELF,* $\texttt{PRF}$ *is a secure PRF, and* $\texttt{IO}$ *is a secure indistinguishability obfuscation, then if H is instantiated as in Construction 7.1 (and the obfuscation is appropriately padded),* $\texttt{SIG}$ *is existentially unforgeable under a chosen message attack*

Proof. We prove security through a sequence of hybrids.

Hybrid 0 This is the standard chosen message attack. Suppose the adversary runs in time s and produces an existential forgery (m^*, σ^*) with non-negligible probability ϵ.

Let $\{m_1, \ldots, m_q\}$ be the messages for which the adversary requested signatures, and let σ_i be the corresponding signature. Let $\widehat{m_i} = f(m_i)$ and $\widehat{m^*} = f(m^*)$.

Hybrid 1 Here, we add a check that $\widehat{m^*} \notin \{\widehat{m_1}, \ldots, \widehat{m_q}\}$. Notice that since f is injective, this is equivalent to $m^* \notin \{m_1, \ldots, m_q\}$, which is already required. Thus this check is redundant and the adversary still succeeds with probability ϵ in **Hybrid 1**.

Hybrid 2 Here, we change f to be lossy. That is, we generate $f \leftarrow \texttt{ELF.Gen}(p(s, 1/(\epsilon/2)))$. By the security of the ELF, **Hybrid 2** is indistinguishable from **Hybrid 1**, except with probability $\epsilon/2$. Thus, the adversary produces an existential forgery with probability at least $\epsilon/2$.

Hybrid 3 Let $r \leq p(s, 1/(\epsilon/2))$ be the image size of f. We modify **Hybrid 2** as follows. Choose a random $\widehat{m^*}$ in the image of f using the efficient enumeration procedure. Now accept the adversary's forgery only if $f(m^*) = \widehat{m^*}$. Notice that $\widehat{m^*}$ is independent of the adversary's view, and therefore $f(m^*) = \widehat{m^*}$ happens with probability at least $1/r$. Thus, the adversary succeeds in **Hybrid 3** with probability at least $\epsilon/2r$.

```
**Inputs:** $m \in \mathcal{M}$
**Constants:** $P, f, g, y^*, \hat{m^*}$
  1. Compute $\widehat{m} \leftarrow f(m)$.
  2. If $\widehat{m} = \hat{m^*}$, output $y^*$ and terminate.
  3. Otherwise, let $x = g(\widehat{m})$.
  4. Output $y = P(x)$.
```

Figure 5: The program $H_{P,f,g,y^*,\hat{m^*}}$.

Hybrid 4 Let y^* be a random point in $\mathcal{X}$ (equivalently, let $y^* = P(x^*)$ for a random $x^* \in \mathcal{X}$). We now replace H with an obfuscation of the program $H_{P,f,g,y^*,\hat{m^*}}$ given in Figure 5.

Suppose our adversary succeeds in forging in **Hybrid 4** with non-negligible advantage. Then we can use such an adversary to invert P. Given a point $y \in \mathcal{X}$, we simulate **Hybrid 4** using $y^* = y$. If the adversary ever queries on a message m such that $\widehat{m} = f(m)$ equals $\widehat{m^*}$, abort the game (since **Hybrid 4** would have rejected anyway). To answer a signing query on message m, simply output $g(f(m))$. It is straightforward to see that, since $\widehat{m} \neq \widehat{m^*}$, this is the correct signature on m. Now the adversary's forgery is a message m^* and signature σ^* such that $P(\sigma) = H(m^*)$ and $f(m^*) = \widehat{m^*}$. Since $f(m^*) = \widehat{m^*}$, we have that $H(m^*) = H_{P,f,g,y^*,\hat{m^*}}(m^*) = y^* = y$. Thus, σ is an inverse of y under P. Therefore we simply output σ. Our success probability is the same as the success probability of the adversary in **Hybrid 4**. It remains to prove that **Hybrid 4** is indistinguishable from **Hybrid 3**, which will complete the security proof.

To that end, endow $\mathcal{Z}$ with a total order. Fix f and let $L = \{z_1, \ldots, z_r\}$ be the sorted list of all r outputs of f (which can be efficiently enumerated by assumption). Let z_0 be smaller than the smallest element in $\mathcal{Z}$. Let i^* be the index such that $z_{i^*} = \widehat{m^*}$. Consider the following hybrids:

Hybrid 3.i for $i = 0, \ldots, r$ This is the same as **Hybrid 3**, except that we generate two PRFs g_1, g_2, and H is an obfuscation of the program $H_{P,f,g_1,g_2,y^*,\hat{m^*},z_i}$ given in in Figure 6

```
**Inputs:** $m \in \mathcal{M}$
**Constants:** $P, f, g_1, g_2, y^*, \hat{m}^*, z$
  1. Compute $\widehat{m} \leftarrow f(m)$.
  2. If $\widehat{m} \leq z$ and $\widehat{m} \neq= \hat{m}^*$, output $y = P(g_1(\widehat{m}))$.
  3. Otherwise, if $\widehat{m} \leq z$ and $\widehat{m} = \hat{m}^*$, output $y^*$ and terminate.
  4. Lastly, if $\widehat{m} > z$, output $y = g_2(\widehat{m})$.
```

Figure 6: The program $H_{P,f,g_1,g_2,y^*,\hat{m^*},z}$.

First notice that $H_{P,f,g_1,g_2y^*,\hat{m^*},z_0}$ is functionally equivalent to $g_2 \circ f$. Thus the obfuscations are indistinguishable. This means **Hybrid 3** is indistinguishable from **Hybrid 3.0**. Similarly, $H_{P,f,g_1,g_2,y^*,\hat{m^*},z_r}$ is functionally equivalent to $H_{P,f,g_2,y^*,\hat{m^*}}$ since all the image points of f are at most z_r. Therefore **Hybrid 3.r** is indistinguishable from **Hybrid 4**.

It remains to show that **Hybrid 3.**$i-1$ is indistinguishable from **Hybrid 3.**i. This follows from a straightforward application of the punctured programming approach of Sahai and Waters [SW14] and the fact that $P(x)$ for a uniform x is identically distributed to a uniform y. We omit the details.

Piecing together, we see that **Hybrid 3** is indistinguishable from **Hybrid 4**, as desired.

Therefore **Hybrid 0** is indistinguishable from **Hybrid 4**, in which the adversary cannot forge. Thus no efficient adversary can forge in **Hybrid 0**. This completes the proof. □

References

[ABG+13] Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry. Differing-inputs obfuscation and applications. Cryptology ePrint Archive, Report 2013/689, 2013. `http://eprint.iacr.org/2013/689`.

[AHKM14] Daniel Apon, Yan Huang, Jonathan Katz, and Alex J. Malozemoff. Implementing cryptographic program obfuscation. Cryptology ePrint Archive, Report 2014/779, 2014. `http://eprint.iacr.org/2014/779`.

[BCP14] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 52–73, San Diego, CA, USA, February 24–26, 2014. Springer, Heidelberg, Germany.

[BFM14] Christina Brzuska, Pooya Farshim, and Arno Mittelbach. Indistinguishability obfuscation and UCEs: The case of computationally unpredictable sources. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 188–205, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany.

[BHK13] Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi. Instantiating random oracles via UCEs. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 398–415, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.

[BP12] Nir Bitansky and Omer Paneth. Point obfuscation and 3-round zero-knowledge. In Ronald Cramer, editor, *TCC 2012: 9th Theory of Cryptography Conference*, volume 7194 of *Lecture Notes in Computer Science*, pages 190–208, Taormina, Sicily, Italy, March 19–21, 2012. Springer, Heidelberg, Germany.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press.

[BST14] Mihir Bellare, Igors Stepanovs, and Stefano Tessaro. Poly-many hardcore bits for any one-way function and a framework for differing-inputs obfuscation. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014, Part II*, volume

8874 of *Lecture Notes in Computer Science*, pages 102–121, Kaoshiung, Taiwan, R.O.C., December 7–11, 2014. Springer, Heidelberg, Germany.

[BST16] Mihir Bellare, Igors Stepanovs, and Stefano Tessaro. Contention in cryptoland: Obfuscation, leakage and UCE. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part II*, volume 9563 of *Lecture Notes in Computer Science*, pages 542–564, Tel Aviv, Israel, January 10–13, 2016. Springer, Heidelberg, Germany.

[BZ13] Dan Boneh and Mark Zhandry. Secure signatures and chosen ciphertext security in a quantum computing world. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 361–379, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.

[Can97] Ran Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In Burton S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 455–469, Santa Barbara, CA, USA, August 17–21, 1997. Springer, Heidelberg, Germany.

[CCR16] Ran Canetti, Yilei Chen, and Leonid Reyzin. On the correlation intractability of obfuscated pseudorandom functions. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part I*, volume 9562 of *Lecture Notes in Computer Science*, pages 389–415, Tel Aviv, Israel, January 10–13, 2016. Springer, Heidelberg, Germany.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *30th Annual ACM Symposium on Theory of Computing*, pages 209–218, Dallas, Texas, USA, May 23–26, 1998. ACM Press.

[GGH$^+$13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th Annual Symposium on Foundations of Computer Science*, pages 40–49, Berkeley, CA, USA, October 26–29, 2013. IEEE Computer Society Press.

[GGHW14] Sanjam Garg, Craig Gentry, Shai Halevi, and Daniel Wichs. On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 518–535, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany.

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 467–476, Palo Alto, CA, USA, June 1–4, 2013. ACM Press.

[GK16] Shafi Goldwasser and Yael Tauman Kalai. Cryptographic assumptions: A position paper. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of*

Cryptography Conference, Part I, volume 9562 of *Lecture Notes in Computer Science*, pages 505–522, Tel Aviv, Israel, January 10–13, 2016. Springer, Heidelberg, Germany.

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *21st Annual ACM Symposium on Theory of Computing*, pages 25–32, Seattle, Washington, USA, May 15–17, 1989. ACM Press.

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd Annual ACM Symposium on Theory of Computing*, pages 99–108, San Jose, California, USA, June 6–8, 2011. ACM Press.

[HJK$^+$14] Dennis Hofheinz, Tibor Jager, Dakshita Khurana, Amit Sahai, Brent Waters, and Mark Zhandry. How to generate and use universal samplers. Cryptology ePrint Archive, Report 2014/507, 2014. http://eprint.iacr.org/2014/507.

[HSW14] Susan Hohenberger, Amit Sahai, and Brent Waters. Replacing a random oracle: Full domain hash from indistinguishability obfuscation. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 201–220, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany.

[ILL89] Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions (extended abstracts). In *21st Annual ACM Symposium on Theory of Computing*, pages 12–24, Seattle, Washington, USA, May 15–17, 1989. ACM Press.

[Nao03] Moni Naor. On cryptographic assumptions and challenges (invited talk). In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 96–109, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Heidelberg, Germany.

[PW08] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 187–196, Victoria, British Columbia, Canada, May 17–20, 2008. ACM Press.

[Rao14] Vanishree Rao. Adaptive multiparty non-interactive key exchange without setup in the standard model. Cryptology ePrint Archive, Report 2014/910, 2014. http://eprint.iacr.org/2014/910.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th Annual ACM Symposium on Theory of Computing*, pages 475–484, New York, NY, USA, May 31 – June 3, 2014. ACM Press.

[Wee05] Hoeteck Wee. On obfuscating point functions. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 523–532, Baltimore, Maryland, USA, May 22–24, 2005. ACM Press.

[Zha12] Mark Zhandry. How to construct quantum random functions. In *53rd Annual Symposium on Foundations of Computer Science*, pages 679–687, New Brunswick, NJ, USA, October 20–23, 2012. IEEE Computer Society Press.

[Zha16] Mark Zhandry. How to avoid obfuscation using witness PRFs. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part II*, volume 9563 of *Lecture Notes in Computer Science*, pages 421–448, Tel Aviv, Israel, January 10–13, 2016. Springer, Heidelberg, Germany.