On Building Fine-Grained One-Way Functions from Strong Average-Case Hardness

Chris Brzuska, Geoffroy Couteau

On Building Fine-Grained One-Way Functions from Strong Average-Case Hardness

Chris Brzuska[1(✉)] and Geoffroy Couteau[2(✉)]

[1] Aalto University, Espoo, Finland
chris.brzuska@aalto.fi
[2] CNRS, IRIF, Université de Paris, Paris, France
geoffroy.couteau@ens.fr

Abstract. Constructing one-way functions from average-case hardness is a long-standing open problem. A positive result would exclude Pessiland (Impagliazzo '95) and establish a highly desirable win-win situation: either (symmetric) cryptography exists unconditionally, or all NP problems can be solved efficiently on the average. Motivated by the lack of progress on this seemingly very hard question, we initiate the investigation of weaker yet meaningful candidate win-win results of the following type: either there are *fine-grained* one-way functions (FGOWF), or non-trivial speedups can be obtained for all NP problems on the average. FGOWFs only require a fixed polynomial gap (as opposed to superpolynomial) between the running time of the function and the running time of an inverter. We obtain three main results:

Construction. We show that if there is an NP language having a very strong form of average-case hardness, which we call *block finding hardness*, then FGOWF exist. We provide heuristic support for this very strong average-case hardness notion by showing that it holds for a random language. Then, we study whether weaker (and more natural) forms of average-case hardness could already suffice to obtain FGOWF, and obtain two negative results:

Separation I. We provide a strong oracle separation for the implication ($\exists$ exponentially average-case hard language $\implies$ $\exists$ FGOWF).

Separation II. We provide a second strong negative result for an even weaker candidate win-win result. Namely, we rule out a black-box proof for the implication ($\exists$ exponentially average-case hard language *whose hardness amplifies optimally through parallel repetitions* $\implies$ $\exists$ FGOWF). This separation forms the core technical contribution of our work.

1 Introduction

In his celebrated 1995 position paper [Imp95], Impagliazzo describes his personal view of the study of average-case complexity, an emergent (at the time) and fundamental area of computational complexity initiated in a seminal work of Levin [Lev86], which aims to characterize NP problems which are not only hard for a worst-case choice of inputs, but also for natural distributions over

O. Dunkelman and S. Dziembowski (Eds.): EUROCRYPT 2022, LNCS 13276, pp. 584–613, 2022.
https://doi.org/10.1007/978-3-031-07085-3_20

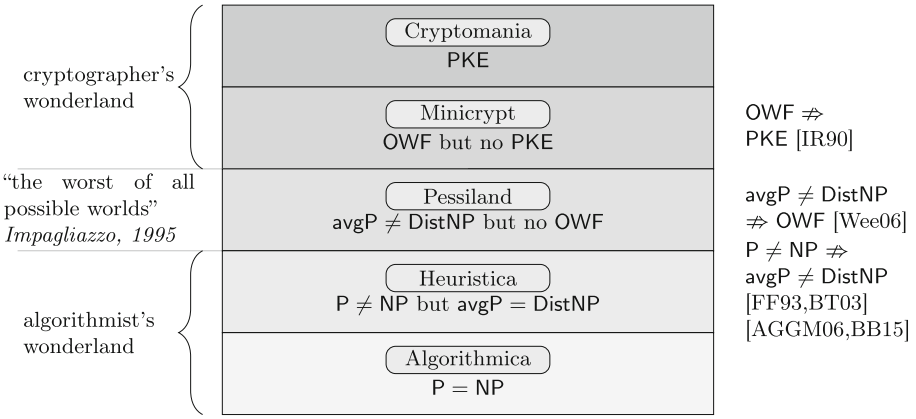

Fig. 1. Impagliazzo's five worlds

the inputs. In Impagliazzo's view, our current understanding of the landscape of complexity theory is best described by considering five possible worlds we might live in, which are now commonly known as the *five worlds of Impagliazzo*, corresponding to the five possible outcomes regarding the existence of worst-case hardness in NP, average-case hardness in NP, one-way function, and public-key cryptography. The corresponding five worlds, Algorithmica, Heuristica, Pessiland, Minicrypt, and Cryptomania, and their relations are summarized on Fig. 1. Algorithmica and Heuristica correspond to the "algorithmist's wonderland", where all NP languages can be decided efficiently on the average. Cryptomania and Minicrypt correspond to the "cryptographer's wonderland",[1] where one-way functions (and therefore, stream ciphers, signatures, pseudorandom functions, etc.) exist. Eventually, Pessiland is what Impagliazzo describes as "the worst of all possible worlds": a world in which many NP problems might be untractable (even on natural instances), yet no one-way function (and thus no cryptography) exists.

One-Way Functions Based on Average-Case Hardness. In this article, we study whether (the existence of) average-case hard NP problems imply (the existence of) one-way functions. Conceptually, a positive answer to this question rules out Pessiland, i.e., it constitutes a win-win result: Either all NP problems can be solved efficiently, on the average, or cryptographic one-way functions exist. Little progress has been made on this question in the past two-and-a-half decades. There is a partial explanation for this lack of success: we know that any construction of one-way functions from average-case hard NP problems must rely on *non-black-box techniques* (Wee [Wee06] attributes this simple observation to Impagliazzo and Rudich). Indeed, similar separations

[1] Though we heard that lately, some cryptographers have been found dreaming of an even higher heaven, the mysterious land of Obfustopia.

[IR90, FF93, BT03, AGGM06, BB15] are known between any two of Impagliazzo's worlds. However, there, the situation is much more satisfying: At the bottom of the hierarchy, we know that stronger *exponential* worst-case assumptions imply that $\mathsf{avgP} \neq \mathsf{DistNP}$ [Yao82, WB86, Sud97, Lev87, STV01]. At the top of the hierarchy, we know that *exponentially secure* one-way functions imply a weak, but useful notion of public-key cryptography, namely *fine-grained* public-key cryptography where there is a polynomial (rather than superpolynomial) gap between the time to encrypt and the time needed to break the cryptosystem [Mer78, BGI08]. Interestingly, the very first publicly-known work on public-key cryptography, the 1974 project proposal of Merkle[2] (published much later in [Mer78]) achieves exactly such a weak notion of security: Merkle shows that an ideal OWF (modeled as a random oracle) can be used to construct a key agreement protocol where the honest parties run in time n, while the best attack requires time n^2. The assumption of an ideal OWF was later relaxed to the existence of exponentially hard OWFs by Biham, Goren and Ishai [BGI08]. Hence, in essence, Merkle establishes a *weak exclusion* of Minicrypt, by showing that strong hardness in Minicrypt already implies some nontrivial form of public-key cryptography, with a quadratic gap between the attacker's runtime and the honest parties' runtime.

1.1 Our Contribution: Inbetween Heuristica and Pessiland

The above result suggests a natural relaxation of Impagliazzo's program: rather than ruling out Pessiland entirely, one could hope to show that sufficiently strong forms of average-case hardness suffice to construct weak forms of cryptography. Such a result would still have a very desirable win-win flavor. For example, if one shows that *exponential* average-case hardness implies *fine-grained* one-way functions, it would show that either *all* NP problems admit nontrivial (subexponential) algorithms on the average, or there must exist *some* form of cryptography, with a polynomial security gap. As the computational power increases, such a gap translates to an increasingly larger runtime gap on concrete instances and thus a larger concrete security margin in realistic situations.

We can also consider starting from even stronger forms of average-case hardness. A very natural target is *non-amortizing* average-case hardness, which states (in essence) that deciding whether k words $(x_1, \cdots, x_k)$ belong to a language $\mathcal{L}$ is k times harder (on average) than deciding membership of a single word. This stronger form of average-case hardness is closely related to the widely studied notion of proofs of work [BRSV18]. Building fine-grained one-way functions from this strong form of average-case hardness would still be a very meaningful win-win: it would show that either we can obtain nontrivial savings on average for all NP problems when amortizing over many instances (which would be an algorithmic breakthrough), or there must exist some weak form of cryptography.

In this work, we initiate the study of these intermediate layers between Heuristica and Pessiland, obtaining both positive and negative results.

[2] Ralph Merkle, 1974 project proposal for CS 244 at U.C. Berkeley, http://www.merkle.com/1974/.

Fine-Grained OWFs from Block-Finding Hardness. We mentioned above two natural strengthenings of average-case hardness: exponential average-case hardness, and non-amortizable average-case hardness (deciding whether k words $(x_1, \cdots, x_k)$ belong to $\mathcal{L}$ is k times harder that deciding membership of a single word). Here, we consider even stronger notions: we assume that there is a language where it is already hard to *decide*, given k random words $(x_1, \cdots, x_k)$, whether their language membership satisfies some local structure. As a simple example of such a notion, consider the following (average-case) *block-finding hardness* notion: given k random words $\vec{x} = (x_1, \cdots, x_k)$ and a $t \approx \log k$-bit string s, computed as the t language membership bits of a randomly chosen sequence of t consecutive words in $\vec{x}$, find these consecutive words. The notion states (informally) that finding such a sequence (when it exists, and with probability significantly better than the random guess) cannot be done much faster than by brute-forcing a significant fraction of all the language membership.

To get an intuition of this problem, it is helpful to consider an even simpler formulation: given k random words $(x_1, \cdots, x_k)$ with the promise that t consecutive words are not in the language, find these t consecutive words (with probability significantly better than $1/k$). For a very hard language, it is not clear how to do this without naively brute-forcing the language membership of $\Omega(k/t)$ words. We show that this (very strong) average-case hardness notion already gets us outside of Pessiland: if there is a block-finding hard language, then there exists fine-grained one-way functions (with a quadratic hardness gap).

Heuristically Evaluating the Assumption on Random Languages. Given that this strong form of average-case hardness is new, we provide some heuristic analysis to support the intuition that it plausibly holds for some hard languages. To do so, we introduce a convenient tool for this heuristic analysis, a *random language model* (RLM), analogous to how the random oracle model [BR93] is used to heuristically study the security of constructions when instantiated with a sufficiently strong hash function. The RLM provides oracle access to a truly random NP-language $\mathcal{L}$. In the RLM, each bitstring $x \in \{0,1\}^n$ belongs to $\mathcal{L}$ with probability exactly $1/2$, and the membership witness for a word $x \in \mathcal{L} \cap \{0,1\}^n$ is a uniformly random bitstring from $\{0,1\}^n$.[3] To check membership to the language, the parties have access to an oracle Chk which, on input a pair $(x, w) \in \{0,1\}^n \times \{0,1\}^n$, returns 1 if $x \in \mathcal{L}$ and w is the right corresponding witness, and 0 otherwise. Finding out whether a random bitstring $x \in \{0,1\}^n$ belongs to $\mathcal{L}$ requires 2^{n-1} calls to Chk on the average.[4]

[3] Of course, this heuristic is simplified: most real languages can have more than a single witness, and the choice of having $|w| = |x|$ is a somewhat arbitrary way of tuning the hardness to make it exactly 2^n. Still, we believe that there is value in using a simple model to heuristically analyze the plausibility of an assumption – even though, as any heuristic model, it must fail on artificial counter examples.

[4] More precisely, it requires 2^{n-1} calls to Chk on the average to find a witness of language membership if x is indeed in the language. In turn, it requires 2^n calls to confirm that there is indeed no witness if x is not in the language.

The RLM captures idealized hard language where it is not only (exponentially) hard to decide language membership, but also hard to sample an element of the language with probability significantly better than 1/2 (hence, in particular, it is also hard to generate a word together with the corresponding witness). This captures hard languages where no further structure is assumed beyond the ability to efficiently check a candidate witness; note that the ability to sample instances together with their witness is exactly the additional structure which implies the existence of one-way functions [Imp95], hence the question of building one-way functions from average-case hardness asks precisely about whether this can be done without assuming this additional structure to start with.

In this work, we prove that a random language satisfies block-finding hardness, providing some heuristic support for this strong form of average-case hardness. Hence, we get as a corollary:

Corollary 1 (Informal). *In the Random Language model, there exists a fine-grained one-way function which can be evaluated with n oracle calls, but cannot be inverted with $o(n^2)$ calls to the random language.*

Constructing a FGOWF from Block-Finding Hardness. At a (very) high level, the construction proceeds as follows: suppose that there exists *hard puzzles* where sampling a random puzzle p is easy (it takes time, say, $O(1)$), but finding the unique solution $s = s(p)$ to the puzzle, and verifying that a candidate solution s to the puzzle is correct, are comparatively harder (they take some much larger respective times N_1 and N_2 with $N_1 \approx N_2$). For example, such puzzles can be constructed by sampling $|s|$ words $(x_1, \cdots, x_{|s|})$, and asking for the length-$|s|$ bitstring of the bits indicating for each word x_i whether it belongs to a given hard language $\mathcal{L}$. Then we construct a fine-grained OWF as follows: an input to the function is a list of n puzzles $(p_1, \cdots, p_n)$ for some bound n, and an integer $i \leq n$. The function $F(p_1, \cdots, p_n, i)$ first solves the puzzle p_i, and outputs the solution $s(p_i)$ together with $(p_1, \cdots, p_n)$. Evaluating F takes time $O(n) + N_1$; on the other hand, when $\mathcal{L}$ is an ideally hard language, inverting F requires brute-forcing many of the p_i, which takes time $O(n \cdot N_2)$. Setting $n \approx N_1 \approx N_2$ gives a quadratic hardness gap. We refer to Sect. 3 for a technical overview and the full version of this work [BC20] for a formal proof and analysis of block-finding hardness in the RLM.

Average-Case Hard Languages and Fine-Grained OWF. With the above, we know that a sufficiently strong form of average-case hardness suffices to construct fine-grained one-way functions. The natural next question is whether weaker forms of average-case hardness could possibly suffice. We consider the two natural notions we mentioned previously: exponential average-case hardness, and the stronger *non-amortizing* exponential average-case hardness. For both, our main results are negative and rule out relativizing (black-box) constructions.

For exponential average-case hard languages, we show that *any* construction of fine-grained OWF from an (even exponentially) average-case hard language,

even with an arbitrarily small polynomial security gap $N^{1+\varepsilon}$ (for any absolute constant $\varepsilon > 0$), must make a non-black-box use of the language. We prove this by exhibiting an oracle relative to which there exists an exponentially hard language, but no fine-grained one-way functions:

Theorem 2 (Informal). *There is an oracle relative to which there exists an exponentially secure average-case hard language, but any candidate fine-grained OWF f can be inverted with probability $O(1)$ and $\tilde{O}(N)$ calls to the oracle, where N denotes the number of oracle calls to compute f in the forward direction.*

Black-Box Separation Between Non-Amortizable Average-Case Hard Languages and Fine-Grained OWF. We then investigate whether non-amortizability (which states, roughly, that deciding membership of k random instances to $\mathcal{L}$ should take $O(k)$ times longer than deciding membership of a single instance to $\mathcal{L}$) suffices to construct fine-grained OWFs. As we explained, this would still constitute a very interesting win-win result: it would show that either weak forms of cryptography exist unconditionally, or nontrivial speedups can be achieved for *all* NP problems *when amortizing over many random instances*. Below, we sketch another motivation for studying this setting.

Non-Amortizability Helps Circumvent Black-Box Impossibilities. Non-amortizability features have proven to be a key approach to overcoming black-box impossibility results for cryptographic primitives. For example, the Biham-Goren-Ishai construction [BGI08] of fine-grained key agreement from exponential OWFs only provides an inverse-polynomial bound on the probability that an attacker retrieves the shared key when relying on Yao's XOR Lemma. In turn, when relying on a (plausible) version of the XOR Lemma stating that success probability decreases *exponentially fast* in the number of XORed instances, the adversary's success probability can be brought down to negligible. Yet, this "Dream XOR Lemma" cannot be proven under black-box reductions [BGI08]. An even more striking example is given by Simon's celebrated black-box separation between one-way functions and collision-resistant hash functions [Sim98]: Holmgren and Lombardi [HL18] recently showed that a one-way *product* function (i.e., a OWF that amplifies twice, meaning that inverting f on two random images (y_1, y_2) takes twice the time of inverting f on a single random image) suffices to circumvent Simon's impossibility result and build a collision-resistant hash function (in a black-box way).

A Black-Box Separation. Motivated by the above, we investigate the possibility of building fine-grained OWFs from non-amortizable average-case hard languages (i.e., languages whose average-case hardness amplifies through parallel repetitions). Unfortunately, our result turns out to be negative: we prove that there is no black-box construction of an $N^{1+\varepsilon}$-hard OWF (where N is the time it takes to evaluate the function in the forward direction), for an arbitrary constant $\varepsilon > 0$, even from an exponentially average-case hard language whose

hardness amplifies at an exponential rate through parallel repetition. Conceptually, our second negative result separates fine-grained one-way functions from a much stronger primitive and can thus be seen as a much stronger result. Note, however, that technically, the two negative results are incomparable since the first one rules out relativizing reductions whereas the latter rules out black-box reductions, see the beginning of Sect. 3 for a discussion.

Theorem 3 (Informal). *There is no black-box construction of an $N^{1+\varepsilon}$-hard OWF f, for an arbitrary constant $\varepsilon > 0$, from exponentially average-case hard languages $\mathcal{L}$ whose hardness amplifies at an exponential rate through parallel repetition.*

In the nomenclature of Reingold, Vadhan and Trevisan [RTV04], we rule out a $\forall\exists$-weakly-reduction, a slightly weaker notion than a relativizing reduction. Namely, the reduction can access the adversary. Our result becomes a full oracle separation if the fine-grained one-way function f would be given black-box access to the adversary $\mathcal{A}$ as well. Reingold, Vadhan and Trevisan point out that in some cases, the adversary $\mathcal{A}$ can be embedded into the oracle O, but doing so did not seem straightforward for our case and is left as an open question. In the CAP nomenclature of Baecher, Brzuska and Fischlin [BBF13], we rule out NNN reductions, since the construction f can depend on the language $\mathcal{L}$, and the reduction $\mathcal{C}$ can depend on both, the adversary and the primitive, i.e., each of these dependencies can be seen as non-black-box, thus $\overline{\text{NNN}}$.

Why Study Non-amortizability? In the past, non-amortizability proved key to overcome related limitations. For example, if one wants to build fine-grained key exchange with overwhelming security from (exponential) OWFs, a strong non-amortizability property (dubbed *dream XOR lemma*) is known to be necessary [BGI08]. Non-amortizability also allows bridging the gap between OWF and CRHFs [HL18]. Moreover, a very natural and—initially—promising-looking approach towards fine-grained cryptography from weaker-than-usual assumptions inherently goes through (and stops at) non-amortizable languages. Finally, a positive result (non-amortizable languages give FGOWF) would give a really nice win-win (algorithms efficiency vs cryptographic security) result.

Let us elaborate on the approach being natural and looking promising. The goal (FGOWFs from "weaker" assumptions) was set forth in [BRSV17], with a promising path: starting from a worst-case assumption (the exponential time hypothesis (ETH)), one gets structured average-case hardness (through the orthogonal vector problem (OV)); furthermore, this was pushed to *non-amortizable* hardness in follow-up work [BRSV18]. Then, [BRSV17] asked: can we push this OV-based construction further, up to FGOWFs?

One way to read our contributions is the following: our *positive* result can easily be framed as an OV-based construction (*solving a block* becomes finding x such that $F(x) = 1$, where F is an explicit low-degree polynomial). The key technique in [BRSV17,BRSV18] are a worst-case to average-case reduction and an average-case to non-amortizability reductions using the Berlekamp-Welch

algorithm, which inherently only works for *search* OV. Block-finding hardness, on the other hand, formalizes what we would *need* to prove to achieve FG-OWF from ETH through this approach. Then, our last separation says: the further the Berlekamp-Welch techniques seems to get us (i.e., to non-amortizing hardness) won't suffice (in a black-box way) to achieve what we need, with *any* candidate construction. In other words, we need new non-black-box techniques that go beyond the Berlekamp-Welch algorithm. Since non-amortizability proved key to overcome related limitations in the past, we actually attempted to *prove* security of our construction from non-amortizing hardness for a long time.

1.2 A Core Abstract Lemma: The Hitting Lemma

At the heart of both our positive result and our black-box separations is an abstract lemma, which we call the *Hitting Lemma*. While the statement of the lemma is very intuitive, its proof is quite technical, and forms one of the core technical contributions of this work. In its abstract form, the Hitting Lemma is a very general probability statement about a simple two-player game between a challenger and an adversary. It shows up naturally on three seemingly unrelated occasions in our work, hence it seems likely that it can have other applications, and we believe it to be of independent interest.

At a high level, the Hitting Lemma provides a strong Chernoff-style bound on the number of witnesses which an adversary can possibly find given oracle access to the relation of a hard language. More precisely, we state the Hitting Lemma in an abstract way, as a game with the following structure:

- First, the game chooses a list of sets V_i. Each set V_i has size bounded by some value 2^n and can be thought of as the set of *candidate witnesses* for a size-n word.
- In each set V_i, the game chooses a uniformly random witness r_i. The sets V_i are allowed to have different sizes, to capture the more general setting where the adversary already obtained preliminary information excluding candidate witnesses.
- Eventually, the adversary interacts with an oracle $\mathsf{Guess}_{r_1 \cdots r_\ell}$ which, on input (i, x), returns 1 if $x = r_i$ and $\perp$ otherwise.

We call a query (i, x) such that $\mathsf{Guess}_{r_1 \cdots r_\ell}(i, x) = 1$ a *hitting query* (or a *hit*). The goal of the adversary is to get as many distinct hits as possible within a bounded number of queries. Intuitively, the most natural strategy to maximize the number of hits is to proceed as follows: first pick the smallest set V_i, and query arbitrary positions one by one, until a hit is obtained. Then, pick the second smallest set V_j and keep proceeding the same way, until all of the r_i are found or the query budget is exhausted.

In essence, the Hitting Lemma states that the above natural strategy is really the best possible strategy, in a strong sense. Namely, denoting m_Q the average number of hits obtained by a Q-query adversary following the above strategy, the Hitting Lemma shows that for *any* possible adversarial strategy, the probability of getting $O(m_Q) + c$ distinct hits using Q queries decreases exponentially

with c (for some explicit constant in the $O(\cdot)$). The proof combines a reduction to a simpler probabilistic statement, proven by induction over Q, with a tight concentration bound on the winning probability of the above natural strategy.

Interestingly, the Hitting Lemma extends directly to the non-uniform setting, where the adversary is allowed to receive an arbitrary k-bit advice about the Guess oracle; this properties turns out to be crucial in some of our results. Our bound shows that this advice cannot provide more than k additional hits. More precisely, for any possible adversarial strategy where the adversary receives an arbitrary k-bit advice about the oracle, the probability of getting $O(m_Q) + k + c$ hits decreases exponentially with c. We refer to Sect. 6 for the full statement and analysis of the Hitting Lemma.

Analogy with the ROM. In the Random Oracle Model, a long line of work (see for example [Hel80, Unr07, DGK17, CDGS18] and references therein) has established the hardness of inverting an idealized random function in a non-uniform setting, given a bounded-length advice about the oracle. These results have proven to be important and powerful tools to reason about the Random Oracle Model. At a high level, the hitting lemma provides a comparable tool in the Random Language Model, and captures the hardness of *deciding language membership* for an idealized hard language, even given a non-uniform advice, and even when the adversary tries to amortize over many instances.

1.3 On the Significance of Our Results

We now address some points about how our results should be interpreted, and what they imply.

On Basing FGOWF on the Average-Case Hardness of a Concrete NP-complete Language. Our impossibility results rule out constructions of fine-grained one-way function that would work using black-box access to an *arbitrary* average-case hard language. However, it seems plausible that a construction of FGOWFs from the average-case hardness of an arbitrary language could proceed differently. Typically, such a construction could first reduce the language to a SAT instance (or any other NP-complete problem) using a non-black-box (e.g. Karp) reduction. Then, the construction of FGOWF would build upon the concrete structure of SAT; such a construction would not be ruled out by our results.

Implications of our Negative Results. In this setting, our negative results should be interpreted as saying that if such a construction is possible, then it must crucially rely on specific structural hardness properties of the chosen language, and not solely on natural properties such as its exponential hardness, or its non-amortizability. As it turns out, this has implications to previous attempts of basing cryptography on weaker hardness assumptions. The work of [BRSV17] showed constructions of fine-grained average-case hard languages from the strong

exponential-time hypothesis (SETH), using a worst-case to average-case reduction based on the orthogonal vector problem. A major open problem left in their work, which they discuss at length, is whether their construction could be strenghtened to give FGOWFs. In a subsequent work [BRSV18], the authors made a step in the right direction, building *proofs of work* from SETH (building upon their result in [BRSV17]). Their construction precisely exploits that due to the specific structure of the orthogonal vector problem, it is possible to show *non-amortizability* of their fine-grained average-case hard language, which suffices to build proofs of work. Our result shows that this non-amortizability does *not* suffice to build FGOWFs: if there is a construction, it must rely on a different structural hardness property.

In fact, we initially designed the construction of FGOWF from block-finding hardness as a construction based on the average-case hard puzzles of [BRSV18], and dedicated an important effort to trying to reduce its security to the non-amortizable average-case hardness of this puzzle, viewing this approach as the most promising direction to based FGOWFs on a worst-case hardness assumption such as SETH. After failing to prove it secure, we realized that our lack of success might be inherent, and turned this realization into a proof by demonstrating the impossibility of basing a FGOWF on non-amortizing hardness in a blackbox way. We hope and believe that our negative results will therefore guide future attempts of basing FGOWFs on weaker assumptions, even attempts that do not ultimately aim at building them from arbitrary hard languages.

Implications of our Positive Results. Furthermore, our *positive* result hints precisely at the type of hardness which could suffice to build FGOWFs: intuitively, what is needed is that the concrete language satisfies some form of *pattern finding hardness*, where given a list of words $(x_1, \cdots, x_n)$, finding whether there is a sub-vector of words whose membership bits (i.e. the vector of bits indicating for each word whether it is in the language) satisfy a given pattern should require deciding membership of a large fraction of all words. This is formalized as *block-finding hardness* in our work (see Sect. 3.1 and the full version of this work [BC20]). We note that other related forms of hardness – where one must decide whether the membership bits of a vector of words satisfy some locally testable property – can also be shown to imply FGOWFs. We note that actually, a very similar type of structural hardness has been used in [LLW19] to build fine-grained one-way functions from concrete average-case hard problems.

1.4 Related Work

Fine-Grained Cryptography. We already pointed out that Merkle's construction [Mer78] provides the first example of fine-grained cryptography (as well as the first known example of public-key cryptography). It was further studied in [BGI08,BM09], and generalized to the quantum setting in [BS08,BHK+11]. Fine-grained cryptography has only become an explicit subject of study recently. The work of [BRSV17,BRSV18] constructs proofs of work from explicit fine-grained average-case hard languages which can be based on the strong

exponential-time hypothesis (SETH), and explicitly poses the problem of building fine-grained one-way functions (while showing some barriers for basing them on SETH via natural approaches). The work of [DVV16] studies a different form of fine-grained cryptography, showing cryptosystems secure against resource-bounded adversaries, such as adversaries in NC_1, under a worst-case hardness assumption. Eventually, the work of [LLW19] is the most closely related to ours: it shows constructions of fine-grained one-way functions and fine-grained encryption schemes from the average-case hardness of concrete problems, such as the Zero-k-Clique problem.

Hardness in Pessiland. While building one-way functions from average-case hardness has remained elusive, some works have investigated other useful forms of hardness which could possibly reside in Pessiland. In particular, in [Wee06], Wee shows that the existence of non-trivial succinct 2-round argument systems for some languages in NP cannot be excluded from Pessiland in a black-box way.

Oracle Techniques. Besides new ideas, our oracle separation relies on several established techniques. We use the *two-oracle* technique of [Sim98, HR04] where one oracle implements the base primitive and the second oracle breaks constructions built from this primitive. As we argue about the *efficiency* of the constructed one-way function, we use similar techniques to Gennaro and Trevisan [GT00] who describe the emulation of a random oracle based on a bounded-length string, implicitly applying a compression argument. We use Borel-Cantelli to extract a single oracle from a distribution of random oracles as the seminal work on black-box separations by Impagliazzo and Rudich [IR89]. In order to make our oracle deterministic, we use the hashing trick of Valiant-Vazirani [VV85] to obtain a unique value out of many pre-image for a one-way function. In particular, we hash *evaluation paths* similar to Bogdanov and Brzuska [BB15] who separate size-verifiable one-way functions from NP-hardness.

On the Relation to Two Recent Works. In a recent work [PV20], Pass and Venkitasubramaniam show that TFNP (the class of total NP search problems) is unconditionally hard in Pessiland. More precisely, they show the following: if there exists average-case hard languages, then either there exists average-case hard TFNP problems, or there exists one-way functions. We note that, since their constructions are black-box, combining their result with our work further implies the following result stating that proving that *total search* average-case hardness suffices to construct fine-grained one-way functions is likely to be hard, since any such black-box proof would unconditionally imply the existence of (full-fledged) one-way functions:

Theorem 4 (this work + [PV20], informal). *If there is a black-box construction of $N^{1+\varepsilon}$-hard one-way function, for an arbitrary constant $\varepsilon > 0$, from average-case* TFNP *hardness, then one-way functions exist unconditionally.*

In another recent work [PL20], Pass and Liu showed that mild average-case hardness of computing time-bounded Kolmogorov complexity already suffices to

establish (in a black-box way) the existence of one-way functions. In particular, we note that, combined with our results, this implies that even exponentially-strong, self-amplifiable average-case hardness in NP does not imply (in a black-box way) mild average-case hardness of time-bounded Kolmogorov complexity.

Theorem 5 (this work + [PL20], informal). *There is no black-box reduction from the mild average-case hardness of computing time-bounded Kolmogorov complexity to the existence of exponentially average-case hard languages whose hardness amplifies at an exponential rate via parallel repetition.*

2 Preliminaries

2.1 Notation, Computational Models and Oracles

For any $n \in \mathbb{N}$, $[n]$ denotes the set $\{1, \cdots, n\}$. Throughout this paper, we represent algorithms as families of boolean circuits (one for each input length), and use circuit size (i.e., the number of wires) as the main measure of efficiency. We model oracle access by allowing circuits to have oracle gates. We measure the size of such an oracle circuit as for a standard circuit, as the number of its wires. Typically, if an oracle takes an n-bit entry as input and outputs an m-bit response, this will be modeled by a fan-in-n fan-out-m oracle gate (hence this gate will contribute $n + m$ to the total circuit size).

As in the standard model for boolean circuits, the wires typically carry bit values. For simplicity and readability, we will generally allow the wires to directly carry other special symbols, such as $\perp$ and err (converting a circuit in this model to a "purely boolean" circuit only introduces some constant blowup which has no impact on our asymptotic results). By default, even when we do not mention it explicitly, we allow all (standard and oracle) gates to receive the symbol err as one of their inputs. If a gate receives err as one of its inputs, it returns the err on all of its output wires. We use pseudo-code as a description language and only argue about the size of the corresponding circuit informally.

2.2 Fine-Grained One-Way Functions

We start by introducing the notion of a fine-grained one-way function (FG-OWF). At a high level, an (ε, δ)-FG-OWF is a function f (modeled as a family $\{f_m\}_m$ of circuits, one for each input size) such that all circuits of size $o(|f|^{1+\delta})$ have probability at most ε to find a preimage of $f(x)$ for a random input x.

Definition 6 (Fine-Grained One-Way Function). *Let $\varepsilon : \mathbb{N} \mapsto \mathbb{R}^+$ be a positive function and $\delta > 0$ be a constant. A function $f : \{0,1\}^* \to \{0,1\}^*$ is an (ε, δ)-fine-grained one-way function if for all circuit families $\mathcal{C} = \{\mathcal{C}_m\}_{m \in \mathbb{N}}$ and all large enough m, if $|\mathcal{C}_m| < |f_m|^{1+\delta}$, then we have*

$$\Pr_{z \leftarrow \$ \{0,1\}^m} \left[\mathcal{C}_m(f(z), 1^m) \in f^{-1}(f(z)) \right] \leq \varepsilon(m).$$

One can also consider a slightly weaker notion, namely a fine-grained one-way *function distribution* (FG-OWFD), were the hardness of inversion should hold with respect to a randomly sampled function f from a distribution D.

Definition 7 (Fine-Grained One-Way Function Distribution). *Let $\varepsilon : \mathbb{N} \mapsto \mathbb{R}^+$ be a positive function and $\delta > 0$ be a constant. A distribution D over functions $f : \{0,1\}^* \to \{0,1\}^*$ is an (ε, δ)-fine-grained one-way function distribution if for all circuit families $\mathcal{C} = \{\mathcal{C}_m\}_{m\in\mathbb{N}}$ and all large enough m, if $|\mathcal{C}_m| < |f_m|^{1+\delta}$ for all f in the support of D, then it holds that*

$$\Pr_{z\leftarrow\$\{0,1\}^m, f\leftarrow\$D}\left[\mathcal{C}_m(f, f(z), 1^m) \in f^{-1}(f(z))\right] \leq \varepsilon(m).$$

Any distribution over FG-OWFs induces a FG-OWFD, but the converse need not hold in general.

2.3 Languages

The class NP contains all languages $\mathcal{L}$ of the form $\mathcal{L} = \{x \mid \exists w, (|w| = \text{poly}(|x|)) \wedge (\mathcal{R}(x, w) = 1)\}$, where $\mathcal{R}$ is a *relation* computable by a polysize uniform circuit. This definition naturally extends to the case where an oracle O is available; in this case, we say that the oracle language $\mathcal{L}^{\mathsf{O}}$ is in NP^{O} if it is of the above form, where $\mathcal{R}$ is computable by a uniform oracle circuit with $|\mathcal{R}| = \text{poly}(|x|)$. When the oracle O is clear from the context, we will sometimes abuse this notation and simply say that the oracle language $\mathcal{L}^{\mathsf{O}}$ is in NP. For a string x, we will denote by $\mathcal{L}(x)$ the bit which is 1 if $x \in \mathcal{L}$, and 0 otherwise. We will also extend this definition to vectors of strings $\vec{x}$ in a natural way.

Average-Case Hard Languages. We now define (exponentially) average-case hard languages (EACHLs). Note that the exponential hardness in the following definition refers to the success probability of the algorithm.

Definition 8 (Exponential Average-Case Hardness). *A language $\mathcal{L}$ is exponentially average-case hard if for any circuit family $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ and all large enough n,*

$$\Pr_{x\leftarrow\$\{0,1\}^n}[\mathcal{C}_n(x) = \mathcal{L}(x)] \leq \frac{1}{2} + \frac{|\mathcal{C}_n|}{2^n}.$$

Note that in the most common definition of EACHLs, one does usually not consider an exact bound $|\mathcal{C}_n|$, and instead define a language to be exponentially hard if a polytime algorithm $\mathcal{C}_n$ finds $\mathcal{L}(x)$ with probability at most $1/2 + \text{poly}(n)/2^n$ for a random word $x \in \{0,1\}^n$. However, since we will work in the fine-grained setting, we settle for a stricter definition, with an explicit relation between the running time of $\mathcal{C}_n$ and the probability of finding $\mathcal{L}(x)$. Similarly as for FG-OWFs, we can also define a weaker notion of exponential average-case hard *language distributions* (EACHLD):

Definition 9 (Exponential Average-Case Hard Language Distribution). *A distribution D over languages $\mathcal{L}$ is exponentially average-case hard if for any circuit family $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ and all large enough n,*

$$\Pr_{x\leftarrow\$\{0,1\}^n,\mathcal{L}\leftarrow\$D}[\mathcal{C}_n(x,\mathcal{L}) = \mathcal{L}(x)] \leq \frac{1}{2} + \frac{|\mathcal{C}_n|}{2^n}.$$

Note that any distribution over EACHLs induces an EACHLD, but the converse need not hold in general.

2.4 Pairwise Independent Hash-Functions

Definition 10. *For all $j, i \in \mathbb{N}$, we call a distribution $\mathcal{H}_{j,i}$ over functions $h : \{0,1\}^j \mapsto \{0,1\}^{i+2}$ a distribution of* pairwise independent *hash-functions, if for all $p, p' \in \{0,1\}^j$ with $p \neq p'$, it holds that*

$$\Pr_{h\leftarrow\$\mathcal{H}_{j,i+2}}\left[h(p) = 0^{i+2}\right] = 2^{-i-2}$$
$$\Pr_{h\leftarrow\$\mathcal{H}_{j,i+2}}\left[h(p') = 0^{i+2}\right] = 2^{-i-2}$$
$$\Pr_{h\leftarrow\$\mathcal{H}_{j,i+2}}\left[h(p) = h(p') = 0^{i+2}\right] = 2^{-2i-4}$$

The following fact is used, e.g., by Valiant and Vazirani in their randomized reduction which solves SAT given a UniqueSAT oracle [VV85].

Claim 1. *For all sets $S \subseteq \{0,1\}^j$ such that $2^i \leq |S| \leq 2^{i+1}$, it holds that*

$$\Pr_{h\leftarrow\$\mathcal{H}_{j,i+2}}\left[\exists! p \in S :\ h(p) = 0^{i+2}\right] \geq \frac{1}{8}.$$

3 Technical Overview: FGOWFs from Block-Finding Hardness

We first introduce the Random Language Model (RLM), which captures idealized average-case hard languages, in the same way that random oracles capture idealized one-way functions.[5] We will use this model as a heuristic tool to analyze the new form of average-case hardness which we will introduce next. We note that this model has limitations: it is a simplified model, and it is actually not too hard to *directly* build a fine-grained OWF in this model (e.g. one can define the function F which maps x to the list of language memberships of the words $x||1, x\cdots, x||n'$, for an appropriate choice of n'[6]). However, such simplified constructions do not correspond to any natural form of average-case hardness that could be formulated on standard NP languages. Rather, our goal is only to use

[5] More formally, since we consider an oracle sampled from a distribution over oracles, as for the Random Oracle Model, this captures average-case hard *language distributions*. I.e., the hardness of a language is averaged over the choice of the instance *and* the sampling of the oracle.

[6] We thank an anonymous reviewer for pointing out this construction.

the RLM as a heuristic rule of thumb to evaluate the plausibility of our new average-case hardness notion.

We define a random language $\mathcal{L}$ as follows: for each integer n and each word $x \in \{0,1\}^n$, sample a uniformly random bit $B[x]$. Then the elements of $\mathcal{L}$ are all x with $B[x] = 1$. For notational convenience, we extend this notation to vectors: given a vector $\vec{x}$ of words, $B(\vec{x})$ denotes the vector of the bits $B[x_i]$. For each $x \in \{0,1\}^n$, we also sample a uniformly random *witness* $W[x] \leftarrow\!\!\$\ \{0,1\}^n$. To check membership to the language, we introduce an oracle Chk defined as follows: on input a pair (x, w), the oracle checks whether $B[x] = 0$ or $w \neq W[x]$. If one of these conditions hold, it outputs $\perp$; otherwise, it outputs 1 (See Fig. 2). It is relatively easy to see that to check membership of a candidate word x to $\mathcal{L}$ given access to Chk, the best possible strategy is to query (x, w) for all possible values $w \in \{0,1\}^n$, hoping to hit the uniformly random value $W[x]$. Hence, deciding membership of a word x to $\mathcal{L}$ requires on the average 2^{n-1} queries to Chk, which shows that $\mathcal{L}$ is (exponentially) average-case hard.

We now define the notion of block-finding hardness. We will show that (1) block-finding hardness holds for a random language, and (2) if there is a block-finding hard language, then there is an explicit construction of a FG-OWF f such that every adversary running in time $N(n)^{2-\nu}$ for an arbitrarily small constant ν has only a negligible probability of inverting f (in n) – *id est*, there exists a $(\mathsf{negl}(n), 1-\nu)$-FG-OWF, where $\mathsf{negl}(n)$ denotes some negligible function of n.

Distribution $\mathcal{T}$	$\mathsf{Chk}[W, B](x, w)$
for $n \in \mathbb{N}$:	**if** $W[x] = w \wedge B[x] = 1$
for $x \in \{0,1\}^n$:	**return** 1
$W[x] \leftarrow\!\!\$\ \{0,1\}^n$	**else return** $\perp$
$B[x] \leftarrow\!\!\$\ \{0,1\}$	
return (W, B)	

Fig. 2. Distribution $\mathcal{T}$ for sampling a random language $\mathcal{L}^{\mathsf{O}} = \{x \in \{0,1\}^* \mid B[x] = 1\}$ with associated list of witnesses W. The oracle $\mathsf{O} = \mathsf{Chk}[W, B]$ allows to check membership of a word $x \in \mathcal{L}^{\mathsf{O}}$ given witness $W[x]$.

3.1 Block-Finding Hardness of $\mathcal{L}$

Informally, we say that a language satisfies *block-finding hardness* if for any adversary $\mathcal{A}$ and any large enough n, the following holds: The adversary $\mathcal{A}$ is given $N \leq 2^n/k$ many length-k vectors $\vec{x_i}$ of distinct words $x_{i,j} \in \{0,1\}^n$ together with the string $s = B[\vec{x_i}]$ (the vector of language membership bits for the words in $\vec{x_i}$) for a uniformly random block index $i \leftarrow\!\!\$\ [N]$. If $\mathcal{A}$ finds the block index i with probability significantly better than guessing, it must run in time $\tilde{\Omega}(N \cdot 2^n)$ (in the RLM, this corresponds to making $\tilde{\Omega}(N \cdot 2^n)$ queries to Chk). Intuitively, this means that (up to polylogarithmic factors) the best strategy to find i is to find out the language membership bits of some of the words in each of the blocks, by brute-forcing every possible witness for these

words, until one finds membership bits that are consistent with s. Slightly more formally, we show the following:

Lemma 11 (Block-Finding Hardness of $\mathcal{L}$ – Informal Version). *For any adversary $\mathcal{C}$, $n \in \mathbb{N}$, block size k, and number of blocks N (with $k \cdot N \leq 2^n$), and any tuple of blocks $(\vec{x_i})_{i \leq N} = (x_{i,1}, \cdots, x_{i,k})_{i \leq N}$ such that all the $x_{i,j}$ are distinct:*

$$\Pr_{i \leftarrow \$[N]}[\mathcal{C}_n((\vec{x_j})_j, B[\vec{x_i}]) = i] \leq \frac{1}{\tilde{O}(N)} \cdot \left(\frac{|\mathcal{C}_n|}{2^n} + 1 \right) \cdot 2^{O(k)}.$$

In the RLM, the language $\mathcal{L}$ satisfies block-finding hardness essentially because distinct words have truly independent witnesses and language membership bits. More formally, the above lemma will follow from a strong and generic concentration bound, the *hitting lemma*. We state and formally prove the hitting lemma separately in Sect. 6, Lemma 19, since it turns out that this lemma provides a very convenient and versatile tool to bound the success probability of an adversary which attempts to decide membership of words in an oracle language (the hitting lemma will be needed on three different occasions in this paper). In the context of proving the block-finding hardness of $\mathcal{L}$, we will need a variant of the hitting lemma of the following form:

Lemma 12 (Simplified Hitting Lemma with Advice – Informal Version). *For every integers $n, N, k \in \mathbb{N}$ with $kN \leq 2^n$, vector $\vec{y}$ of kN words, adversary $\mathcal{A}$ getting $\vec{y}$ and $B[\vec{y}_i]$ for a random i (where $\vec{y}_i$ is a vector of k words), and for every integer $c \geq 1$,*

$$\Pr_{(W,B) \leftarrow \mathcal{T}} \left[\#\mathsf{Hit} \geq \frac{O(|\mathcal{A}|)}{2^n} + k + c \right] \leq 2^{-O(c)},$$

where $\#\mathsf{Hit}$ counts the number of witnesses found by $\mathcal{A}$ for distinct words of length n among the entries of $\vec{y}$.

At the same time, conditioned on making less than M hits in different blocks, it is straightforward to show that $\mathcal{A}$ can find i with probability M/N: intuitively, this is because if i belongs to one of the $N - M$ blocks where no hits were made, then the indices of all these blocks are perfectly equiprobable conditioned on the view of $\mathcal{A}$. Applying Bayes rule to combine the above bounds, the probability that $\mathcal{A}$ finds i is upper bounded by the probability that $\mathcal{A}$ finds i conditioned on making less than M hits, plus the probability of making more than M hits. Therefore, for any M, the probability that $\mathcal{A}$ finds i is upper bounded by

$$\frac{M}{N} + 2^{-O(M - |\mathcal{A}|/2^n - k)}.$$

From there, an appropriate choice of M (depending on $|\mathcal{A}|, N$, and n) suffices to conclude that $\mathcal{A}$ finds i with probability at most $\frac{1}{\tilde{O}(N)} \cdot \left(\frac{|\mathcal{C}_n|}{2^n} + 1 \right) \cdot 2^{O(k)}$, which concludes the proof.

From Block-Finding Hardness to Fine-Grained One-Way Function. A block-finding hard language immediately leads to a FG-OWF with a quadratic hardness gap: the input to the function is a list of $N = 2^n/k$ blocks $\vec{x}$ of distinct words $\vec{x_i}$ together with an index i. Evaluating the function is done by brute-forcing the languages membership bits of the words in $\vec{x_i}$, which takes at most $k \cdot 2^n$ queries to Chk, and outputting $(\vec{x}, s = B[\vec{x_i}])$. By the block finding hardness of $\mathcal{L}$, inverting the function on a random input, on the other hand, requires $\tilde{O}(N \cdot 2^n) = \tilde{O}(2^{2n}/k)$ queries to Chk to succeed with constant probability when the index i is uniquely defined (i.e., there is a unique index i such that the block $\vec{x_i}$ satisfies $s = B[\vec{x_i}]$). This can be guaranteed to hold except with negligible probability, by choosing $k = \omega(\log n)$. Overall, this leads to a FG-OWF with quadratic hardness gap (up to polylogarithmic factors), with some small but non-negligible inversion probability ε. Parallel amplification can then be used to make the inversion probability negligible, leading to the following corollary:

Corollary 13. *For any $\varepsilon > 0$, there exists a $(\mathsf{negl}(m), 1 - \varepsilon)$-fine-grained one-way function distribution in the Random Language Model.*

4 Overview: No FGOWFs from Average-Case Hardness

Next, we study the possibility of instantiating the above construction using an average-case hard language, instead of a block-finding hard language. At first sight, it is not clear that average-case hardness suffices, since our construction crucially relies on the block-finding hardness of the language, a seemingly much stronger property. Indeed, we show that there exists no construction of essentially any non-trivial FG-OWF making a black-box use of an exponentially average-case hard language. To do so, we exhibit an oracle distribution relative to which there is an exponentially average-case hard language, but no FG-OWF, even with arbitrarily small hardness gap. This proof is the only part of our paper that does not require the hitting lemma.

Language Description. We start by introducing our language. Our oracle defines a somewhat exotic language: for each integer k, we let *all* words $x \in \{0,1\}^n$ such that $k = \lceil \log n \rceil$ have the *same* random witness $w \leftarrow\!\!\$\ \{0,1\}^{2^k}$, and we put either all these words simultaneously inside or outside the language, by picking the same random membership bit b_k for all of them. Intuitively, this provides an extreme example of a language which is still hard to decide (since given a word $x \in \{0,1\}^n$, one must still enumerate over $2^{2^{\lceil \log n \rceil}} > 2^n$ candidate witnesses to find out whether $x \in \mathcal{L}$), but whose hardness does not amplify at all (since finding a witness for a single word x gives the witness for all words whose bitlength is close to that of x). This aims at capturing the intuition that any candidate FG-OWF built from an average-case hard language $\mathcal{L}$ must somehow leverage some amplification properties of the hardness of $\mathcal{L}$. Then, the oracle Chk is similar as before: on input (x, w), it returns $\perp$ if $x \notin \mathcal{L}$ or w is not the right witness for x, and 1 otherwise. We will show that any oracle adversary $\mathcal{A}$

requires $O(2^n)$ queries to decide membership of a word $x \in \{0,1\}^n$ to $\mathcal{L}$. The proof is relatively straightforward and relies on the fact that the membership of x to $\mathcal{L}$ remains random conditioned on the view of $\mathcal{A}$ as long as $\mathcal{A}$ did not make any hit, i.e., a query with the right witness for x.

Inexistence of FG-OWF Relative to Chk. Next, we show that for any constant δ, there exists an oracle algorithm $\mathcal{A}$ such that for any candidate FG-OWF f, $\mathcal{A}$ (given access to Chk) of size bounded by $|f|^{1+\delta}$ which inverts f with probability 0.99. The adversary works as follows: for any integer k, it checks whether the function will make "too many" queries of the form (x, w) with x of length n such that $k = \lceil \log n \rceil$ (we call this a k-query), where "too many" is defined as $(2^{2^k})^\varepsilon$ for a value $\varepsilon = (1+\delta/2)^{-1}$. Intuitively, making more than this number of queries ensures that f will have a noticeable probability of making a hitting query. For all such "heavy queries", $\mathcal{A}$ makes all possible (2^{2^k}) queries to Chk with respect to some fixed word x, until he finds the witness. $\mathcal{A}$ also does the same for all k-queries with $k \leq B(\varepsilon)$ for some bound $B(\varepsilon)$ to be determined later, even when they do not correspond to heavy query (this is to avoid some "border effects" of small queries in the probability calculations). Note that this allows $\mathcal{A}$ to find the witness for *all* words of length n such that $k = \lceil \log n \rceil$, since they all share the same witness. $\mathcal{A}$ defines the following oracle-less function f' that contains all the hardcoded witnesses that $\mathcal{A}$ recovered. Now, on input x, f' runs exactly as f and if f makes a k-query (x, w) for some k, then f' proceeds as follows:

- If k corresponds to a heavy query, then, using (2^{2^k}) queries, $\mathcal{A}$ already computed the witness for all k-queries and thus f' contains the hardcoded witness to correctly answer the query.
- If k does not correspond to a heavy query, f' simulates the answer of the oracle as $\perp$.

We prove that with high probability (at least 0.999), the function f' agrees with f on a random input x; this is because f' disagrees with f only if there is a k-query with $k > 10$ where f makes less than $(2^{2^k})^\varepsilon$ queries, yet hits a witness (for all other types of queries, $\mathcal{A}$ finds the witness by brute-force, hence it can always simulate correctly the answer of the oracle). But this happens only with probability $1 - \sum_{k=B(\varepsilon)+1}^{\infty} (2^{2^k})^\varepsilon \cdot 2^{-2^k}$, which is bounded by 0.999 by picking a sufficiently large bound $B(\varepsilon)$ such that $(1-\varepsilon)2^{B(\varepsilon)} > B(\varepsilon)$. Then, by a straightforward probability calculation, the probability that inverting f' (which $\mathcal{A}$ can easily do locally since f' is oracle-less) corresponds to successfully inverting f on a random input x can be lower-bounded by $0.999^2 > 0.99$, which concludes the proof.

5 Overview: No FG-OWF from Non-Amortizable Hardness

Note that the techniques from our simpler oracle separation crucially exploit that the hardness of the average-case hard language implemented by Chk does

not amplify well (in fact, this is the reason why the hitting lemma is not needed in the analysis). We are thus interested in understanding whether we can still provide a black-box impossibility result even when the underlying average-case hard language satisfies *non-amortizable* exponential hardness, or whether non-amortizable average-case hard languages suffice to construct a fine-grained one-way function.

We call a language $\mathcal{L}$ (exponentially) self-amplifiable average-case hard if for any superlogarithmic (computable, total) function $\ell(\cdot)$, for any circuit family $\mathcal{C} = \{\mathcal{C}_n : \{0,1\}^{\ell(n)\cdot n} \mapsto \{0,1\}^n\}_{n\in\mathbb{N}}$ of size at most $2^{O(n)} \cdot \ell(n)$, and for all large enough $n \in \mathbb{N}$,

$$\Pr_{x\leftarrow\$\{0,1\}^{\ell\cdot n}}[\mathcal{C}_n(\vec{x}) = \mathcal{L}(\vec{x})] \leq \mathsf{poly}(n) \cdot 2^{-\left(\ell(n) - \frac{\tilde{O}(|\mathcal{C}_n|)}{2^{O(n)}}\right)}.$$

Informally, this means that to find the language membership bits of $\ell(n)$ challenge words, the best an adversary $\mathcal{C}_n$ can do (up to polylogarithmic factors in $|\mathcal{C}_n|$ and constant factors in n) is to brute-force as many membership bits as it can (roughly, $\tilde{O}(|\mathcal{C}_n|)/2^n$ since brute-forcing a single membership bit requires $O(2^n)$ queries), and guessing the $\ell(n) - \tilde{O}(|\mathcal{C}_n|)/2^n$ missing membership bits at random. Note that self-amplifiable average-case hardness is especially interesting when the circuit $\mathcal{C}_n$ is allowed to run in time larger than 2^n (for small circuits, of size much smaller than 2^n, the standard average-case hardness notion already bounds their probability of guessing correctly a single entry of $\mathcal{L}(\vec{x})$). In this range, the $\mathsf{poly}(n)$ factor in our definition is absorbed in the $\tilde{O}(|\mathcal{C}_n|)$ term in the exponent (note also that adversaries of size larger than $2^{O(n)} \cdot \ell(n)$ can solve the full challenge by brute-force).

Our main result rules out black-box reductions from any exponentially self-amplifiable average-case hard language to fine-grained one-way functions, with arbitrarily small hardness gap. Slightly more formally, we prove the following theorem:

Theorem 14 (Informal). *There exists an oracle* O *and an oracle language* $\mathcal{L}^{\mathsf{O}}$ *such that for any fine-grained one-way function* f*, there exists an (inefficient) adversary* $\mathcal{A}$ *that inverts* f *with probability close to* 1 *such that* $\mathcal{L}$ *remains exponentially self-amplifiable average-case hard against any candidate reduction* $\mathcal{C}$ *given oracle access to both* O *and* $\mathcal{A}$*.*

We prove Theorem 14 which is phrased in terms of reductions by establishing Theorem 15 which is phrased in terms of oracle worlds.

Theorem 15 (Language Hardness and Good Inversion, Informal). *There exists an oracle* O *and an oracle* Inv *such that for all oracle functions* f*, there exists an inverter* $\mathcal{A}$ *of size* $|\mathcal{A}| = \tilde{O}(|f|)$ *which, given oracle access to* $(\mathsf{O}, \mathsf{Inv})$ *and input* (f, y)*, outputs a preimage of* y *with respect to* f^{O} *with probability close to* 1*. Moreover, there exists an oracle language* $\mathcal{L}^{\mathsf{O}}$ *which is exponentially self-amplifiable average case hard against any candidate reduction* $\mathcal{C}$ *given oracle access to* $(\mathsf{O}, \mathsf{Inv})$*.*

Theorem 15 is slightly different from our main theorem: the inverter $\mathcal{A}$ is now required to be efficient, but gets the help of an additional oracle Inv. Furthermore, the reduction $\mathcal{C}$ is now given oracle access to $(\mathsf{O}, \mathsf{Inv})$ instead of $(\mathsf{O}, \mathcal{A})$; the implication follows from the fact that the code of $\mathcal{A}$ is linear in its input size, and thus, its code can be hardcoded into the code of $\mathcal{C}$, hence the reduction $\mathcal{C}^{\mathsf{O},\mathcal{A}}$ in our main theorem can be emulated by a reduction $\mathcal{C}_{\mathcal{A}}^{\mathsf{O},\mathsf{Inv}}$ in Theorem 15, where $|\mathcal{C}_{\mathcal{A}}| \approx |\mathcal{C}|$. To prove Theorem 15, we rely on a standard method in oracle separations: we first prove a variant of Theorem 15 with respect to a *distribution* over oracles O, Inv (where both the success probability of the inverter and the probability of breaking the self-amplifiable average-case hardness of $\mathcal{L}$ will be over the random choice of O, Inv as well). Then, we apply the Borel-Cantelli lemma to show that with measure 1 over the choice of the oracle, the oracle is "good" and thus, in particular, a single good oracle exists as required by Theorem 15. In summary, to prove Theorem 15 we prove two theorems relative to an explicit distribution $\mathcal{T}$ over oracles O, Inv:

Theorem 16 (Language Hardness, Informal). *For any $\ell : \mathbb{N} \mapsto \mathbb{N}$, circuit family $\mathcal{C} = \{\mathcal{C}_n\}_n$, and for all large enough $n \in \mathbb{N}$,*

$$\Pr_{\vec{x} \leftarrow \$ \{0,1\}^{\ell \cdot n}, (\mathsf{O},\mathsf{Inv}) \leftarrow \$ \mathcal{T}} \left[\mathcal{C}_n^{\mathsf{O},\mathsf{Inv}}(\vec{x}) = \mathcal{L}^{\mathsf{O}}(\vec{x}) \right] \leq \mathsf{poly}(n) \cdot 2^{-\left(\ell(n) - \frac{\tilde{O}(|\mathcal{C}_n|)}{2^{O(n)}}\right)}.$$

Theorem 17 (Efficient Inversion, Informal). *Let $f : \{0,1\}^* \to \{0,1\}^*$ be an oracle function. There exists an efficient inverter $\mathcal{A}^{\mathsf{O},\mathsf{Inv}}(f, .)$ for f. More precisely, $\mathcal{A}$ is of size $|\mathcal{A}| = \tilde{O}(|f|)$ and for sufficiently large $m \in \mathbb{N}$, it holds that*

$$\Pr_{z \leftarrow \$ \{0,1\}^m, (\mathsf{O},\mathsf{Inv}) \leftarrow \$ \mathcal{T}} \left[f^{\mathsf{O}}(\mathcal{A}^{\mathsf{O},\mathsf{Inv}}(f, f^{\mathsf{O}}(z))) = f^{\mathsf{O}}(z) \right] \approx 1.$$

5.1 Defining the Oracle Distribution $\mathcal{T}$

The distribution $\mathcal{T}$ samples a triple (W, B, H) where:

- B defines a random language $\mathcal{L}$: for every $x \in \{0,1\}^*$, $B[x]$ is set to 0 or 1 with probability 1/2;
- W defines a set of random *witnesses*: for any $n \in \mathbb{N}$ and $x \in \{0,1\}^n$, $W[x]$ is set to a uniformly random bitstring w_x of length n.
- H contains a pairwise independent hash-function for each triple (i, C, y), where $i \in \mathbb{N}$, C is an encoding of a circuit and y is a bitstring.

A sample (W, B, H) from $\mathcal{T}$ defines a pair of oracles $(\mathsf{O}, \mathsf{Inv})$, where the oracle $\mathsf{O} = (\mathsf{Chk}, \mathsf{Pspace})$ is defined as follows:

- Chk is a membership checking oracle: on input (x, w), it returns $\perp$ if $W[x] \neq w$, and $B[x]$ otherwise. Note that this means that relative to Chk, $\mathcal{L}$ is a random language in $\mathsf{NP} \cap \mathsf{co\text{-}NP}$, since Chk allows to check both membership and non-membership in $\mathcal{L}$, given the appropriate witness. A *hit* is a query to Chk which does not output $\perp$. To emphasize the dependency of $\mathcal{L}$ on O, we use the notation $\mathcal{L}^{\mathsf{O}}$.

– Pspace is a PSPACE oracle which allows the caller to efficiently perform computations that do not involve calls to the oracles $\mathsf{Chk}, \mathsf{Inv}$.

We now turn our attention to the oracle Inv, which is the most involved component: Inv must be defined such that there is an efficient oracle algorithm $\mathcal{A}$ which can, given access to O, Inv, invert any candidate one-way function f^{O}, yet no algorithm (reduction) can break the self-amplifiable average-case hardness of the language $\mathcal{L}^{\mathsf{O}}$ given access to O, Inv. Hence, the goal of Inv is, given an input (f, y), to help compute preimages z of y with respect to the oracle function f^{O}, but with carefully chosen safeguards to guarantee that Inv cannot be abused to decide the language $\mathcal{L}^{\mathsf{O}}$. Our solution relies on two crucial safeguards, which we describe below.

First Safeguard: Removing Heavy Paths. The oracle Inv refuses to invert functions f on outputs y if the query-path from the preimage z to y in f^{O} is "too lucky" with respect to O. To understand this, consider the following folklore construction of a worst-case one-way function f: on input (x, w), it queries $\mathsf{Chk}(x, w)$ and outputs $(x, 1)$ if the check succeeds, and $(x, 0)$ otherwise. Then, querying Inv on input $(f, (x, 1))$ allows the adversary to find the witness w associated to x efficiently, since the function f makes only a single query and thus the inversion query $\mathsf{Inv}(f, (x, 1))$ has small cost for $\mathcal{A}$.

But since f^{O} is a normal (average-case) one-way function, we can allow the oracle to not invert on a too lucky evaluation path, if we can show that it still inverts sufficiently often. Concretely, on input (f, y), the oracle Inv computes the set S of all *paths* from an input z to $y = f^{\mathsf{O}}(z)$, defined as the sequence of input-output pairs. Then, for all $k \leq |f|$, Inv discards from this set S all *k-heavy paths*, i.e., the paths along which the number of Chk hits on k-bit inputs is much higher[7] than expected, i.e., $N(k)/2^{k-1}$, where $N(k)$ is the number of Chk gates with k-bit inputs in f.

If S is not empty, then Inv samples a uniformly random element from S and returns the set of queries made on the path to the adversary. Since oracles need to be deterministic, we derandomize the sampling via the use of the pairwise independent hash-function stored in the third output H of $\mathcal{T}$ at $H[\log|S|, f, y]$ by the Valiant-Vazirani [VV85] trick that ensures that with probability $\frac{1}{8}$, there is only a unique value in S that hashes to $0^{\log(|S|-1)}$. Note that it suffices to return the *set* of query-answer pairs, as the adversary can use the Pspace oracle to find an input z that leads to y with this set of query-answer pairs produced by f^{O}. I.e., the Pspace uses the set to emulate the answers to queries made by f and discards a candidate z as soon as it makes a query not in the set.

Let us return to the issue of k-lightness. Firstly, note that we need to check for lightness for all values k, since the oracle Inv accepts functions that make queries to Chk on different k-values, and the Inv-oracle does not "know" the length of the x_i-values for which $\mathcal{C}$ tries to decide membership. Secondly, we now need to clarify that we consider the number of hits as too high above its expected value

[7] Determining an appropriate bound on *much higher* is crucial to avoid that deciding $\mathcal{L}^{\mathsf{O}}$ becomes too easy. We return to this issue shortly.

if there are more than $O(N(k))/2^k + \log^2(|f|)$ k-hits on the evaluation path. In this case, if $|f| = O(2^k)$, then on input length k, the adversary could essentially get the same number of hits without Inv queries by using a circuit of slightly bigger size $\tilde{O}(|f|)$ that only makes Chk queries. The point of the additive $\log^2 |f|$ term is to ensure (via a concentration bound) that on a uniformly random input z, the probability that the path on z is light is at least $1 - \frac{1}{\mathsf{superpoly}|f|}$ (while at the same time, the language hardness is maintained).

In turn, when $|f|$ is smaller than, say, $2^{\frac{k}{6}}$, then the additive $\log^2 |f|$ term turns out to allow for too many hits. In this case, the probability of making even a single hit is $2^{-\frac{5(k-1)}{6}}$ and thus exponentially small in k whereas $O(N(k))/2^k + \log^2 |f|$ might potentially allow for many hits. Thus, before performing all steps described in the first saveguard, we first replace f by a shaved function f_s, described below.

Second Safeguard: Shaving High Levels. We *shave* all Chk-gates of $|f|$ that are for *large* input length k, i.e., for all Chk-gates with input length k such that $|f| \leq 2^{\frac{k}{6}}$. To do so, we replace f by a shaved function f_s where the answers of such Chk queries are hardcoded to be $\perp$. The probability (over O and z) that this changes the behaviour of f is equal to the probability of making a hit on one of these high levels and thus $2^{-\frac{5(k-1)}{6}}$ for the smallest k such that $|f| \leq 2^{\frac{k}{6}}$, i.e., $k \geq 6\log(|f|)$. Thus, $2^{-\frac{5(k-1)}{6}} \leq m^{-3}$, where $m = |z|$. Note that later, in the Borel-Cantelli Lemma, we need to sum over these bad events, and thus, it is important that the sum of m^{-3} over all m is a constant.

Putting Everything Together. Finally, with the above two safeguards, our oracle Inv works as follows: on input (f, y), it first shaves f of its higher-level Chk gates, computing $f_s \leftarrow \mathsf{shave}(f)$. Then, it constructs the set S of all paths from some input z to $y = f_s^{\mathsf{O}}(z)$, where a path is defined to be the set of all query pairs to O made during the evaluation of f_s on z. Afterwards, it removes from S all paths which are too heavy, where a path is called heavy if there is a k such that it contains a number $N(k)$ k-Chk queries, out of which more than $O(N(k))/2^k + \log^2|f|$ are hits. Eventually, it returns a path from this set S of light paths using the hashing trick to derandomize the sampling.

As we already outlined, the last output H of $\mathcal{T}$ is therefore a set which contains, for every possible triple (i, f, y) where i is an integer, f is an oracle function, and y is a bitstring, a hash function $h = H[i, f, y]$. The guarantee offered by h is that for any set S' of size $2^{i-1} \leq |S'| \leq 2^i$, the probability of the random choice of $h = H[i, f, y]$ that S' contains exactly one entry s such that $h(s) = 0$ is at least $1/8$. Hence, after it computes the set S of light paths, Inv compute the unique integer i such that $2^{i-1} \leq |S| \leq 2^i$, retrieves $h \leftarrow H[i, f, y]$, and output the unique path $p \in S$ such that $h(p) = 0$, or $\perp$ if there is no unique such path. Note that this oracle Inv can fail to return a valid path from an input z to the target output y in f for three reasons: because shaving caused f_s to differ from f on input z (we show that this is uniquely for a random z), because the path from z to y is heavy (again, we show that this is unlikely), and because there is not a unique $p \in S$ such that $h(p) = 0$ (but with probability at least $1/8$, there will be a unique such p). This last source of failure can be later

removed by a straightforward parallel amplification, by querying Inv on many pairs (f_k, y) where the f_i are functionally equivalent variants of f (in which case the corresponding $h_k = H[i, f_k, y]$ are independently random by construction). Note that we could have also hardcoded "true" randomness into Inv instead of using the hashing trick. However, as we will see, the hashing trick enables a compression argument since (a) the hash-functions are sampled *independently* from W and B and (b) the sampling can be emulated when only knowing a single element in the set as well as the size of the set S. Details follow in the next section.

5.2 Proving Theorem 16

Fix a function $\ell : \mathbb{N} \mapsto \mathbb{N}$, a circuit family $\mathcal{C}$, and an integer $n \in \mathbb{N}$. We want to bound the probability, over the choice of $\vec{x} \leftarrow\!\!\$\ \{0,1\}^{\ell(n)\cdot n}$ and $(\mathsf{O}, \mathsf{Inv}) \leftarrow\!\!\$\ \mathcal{T}$, that $\mathcal{C}_n^{\mathsf{O},\mathsf{Inv}}(\vec{x}) = \mathcal{L}^{\mathsf{O}}(\vec{x})$. We proceed in two steps:

- First, we prove an *emulation lemma* which states that there is an explicit algorithm $\mathsf{Emu}^{\mathsf{O}}$ which emulates $\mathcal{C}_n^{\mathsf{O},\mathsf{Inv}}$ without calling the oracle Inv, but using instead some partial information $g(W, B, H)$ about (W, B, H). By emulating, we mean that $\mathsf{Emu}^{\mathsf{O}}(\vec{x}, g(W, B, H)) = \mathcal{C}_n^{\mathsf{O},\mathsf{Inv}}(\vec{x})$, and Emu makes the same number of queries to O as $\mathcal{C}_n$.
- Second, we use the *hitting lemma*, which we already mentioned in Sect. 3 (in the technical overview about the existence of $\mathsf{FG\text{-}OWFs}$ in the RLM), to bound the number of hits on $\vec{x}$ that Emu can possibly make (where a hit on $\vec{x}$ is a query of the form $(x_i, W[x_i])$ to Chk, from which Emu learns whether $x_i \in \mathcal{L}^{\mathsf{O}}$).

The Emulation Lemma. Concretely, we give an explicit algorithm Emu such that $\mathsf{Emu}^{\mathsf{O}}(L, \vec{x}, \mathcal{C}_n) = \mathcal{C}_n^{\mathsf{O},\mathsf{Inv}}(\vec{x})$ and Emu makes the same queries to O as $\mathcal{C}_n$, where the leakage string L contains the following information:

- The sets H and $(W_{\vec{x}}, B_{\vec{x}})$ of all witnesses and membership bits except for those corresponding to the entries of $\vec{x}$ (intuitively, this corresponds to giving to Emu all information about Inv which is sampled independently of the $W[x_i], B[x_i]$ and does not help with finding $\mathcal{L}^{\mathsf{O}}(\vec{x})$).
- The sets $(W^{\mathsf{Hit}}, B^{\mathsf{Hit}})$ which contains all Chk-hits on $\vec{x}$ in paths obtained by $\mathcal{C}_n$ through queries to Inv.
- The set $W^{\overline{\mathsf{Hit}}}$ which contains all other (non-hitting) Chk-query pairs in paths obtained by $\mathcal{C}_n$ through queries to Inv.
- A list I which for each query (f, y) of $\mathcal{C}_n$ to Inv indicates whether this query returned $\perp$ or not, and if it did not, the value i which was used to select the hash function $h = H[i, f, y]$.

The emulation proceeds by using its information: Emu runs $\mathcal{C}_n$ internally on input $\vec{x}$, forwarding its queries to O. Each time $\mathcal{C}_n$ makes a query (f, y) to Inv, Emu first retrieves from I the information whether Inv outputs $\perp$ or not. If it does not, Emu tries all possible inputs z to f^{O}, but without actually querying O:

for each possible input z, Emu runs $f^{\mathsf{O}}(z)$ by retrieving the answers of O from the sets $(W_{\vec{x}}, B_{\vec{x}}, W^{\mathsf{Hit}}, B^{\mathsf{Hit}}, W^{\overline{\mathsf{Hit}}})$. If $f^{\mathsf{O}}(z)$ makes a query whose answer is not contained in these sets or if $f^{\mathsf{O}}(z)$, Emu discards candidate z.

After trying all inputs to f, Emu has a set S' of candidate inputs z, with a corresponding path. Then, it retrieves the index i from I and selects $h \leftarrow H[i, f, y]$, and sets the output of Inv on (f, y) to be the unique path p associated to some $z \in S'$ such that $h(p) = 0$; by construction, there will be a unique such path. The correctness of the emulation follows by construction and by definition of the sets $(W_{\vec{x}}, B_{\vec{x}}, W^{\mathsf{Hit}}, B^{\mathsf{Hit}}, W^{\overline{\mathsf{Hit}}})$ which Emu gets as input.

This emulation highlights the rationale behind the design of Inv: the use of a hash function h to select the output guarantees that, on top of the sets $(W_{\vec{x}}, B_{\vec{x}}, W^{\mathsf{Hit}}, B^{\mathsf{Hit}}, W^{\overline{\mathsf{Hit}}})$, Emu will only need to receive a *relatively small* amount of additional "leakage", corresponding to the list of all values i for each query to Inv. Now, by definition, i is at most $\log |S|$, where S is a set of paths in f, hence $|S| \leq 2^{|f|}$. Therefore, $i \leq |f|$, hence i can be represented using at most $\log |f|$ bits. By construction, a query (f, y) to Inv can leak information about $\vec{x}$ only if $|f| \geq 2^{n/C}$, because otherwise all n-Chk gates gets removed by $\mathsf{shave}(f)$. Hence, our emulator gets a total amount of leakage about $\vec{x}$ bounded by $|\mathcal{C}_n|/2^{O(n)}$. From there, we want to prove that

$$\Pr_{\vec{x} \leftarrow\$ \{0,1\}^{\ell \cdot n}, (\mathsf{O}, \mathsf{Inv}) \leftarrow\$ \mathcal{T}} \left[\mathcal{C}_n^{\mathsf{O}, \mathsf{Inv}}(\vec{x}) = \mathcal{L}^{\mathsf{O}}(\vec{x}) \right] \leq \mathsf{poly}(n) \cdot 2^{-\left(\ell(n) - \frac{\tilde{O}(|\mathcal{C}_n|)}{2^{O(n)}}\right)}.$$

We will do so by proving that

$$\Pr_{\vec{x} \leftarrow\$ \{0,1\}^{\ell \cdot n}, (\mathsf{O}, \mathsf{Inv}) \leftarrow\$ \mathcal{T}} \left[\mathsf{Emu}^{\mathsf{O}}(L, \vec{x}, \mathcal{C}_n) = \mathcal{L}^{\mathsf{O}}(\vec{x}) \right] \leq \mathsf{poly}(n) \cdot 2^{-\left(\ell(n) - \frac{\tilde{O}(|\mathcal{C}_n|)}{2^{O(n)}}\right)}. \tag{1}$$

Bounding Eq. 1 is the goal of the *hitting lemma*.

Applying the Hitting Lemma. The hitting lemma states that for any circuit $\mathcal{C}_n$, any algorithm $\mathcal{A}$ having only access to the inputs and oracles of $\mathcal{C}_n$'s emulator (i.e., $\mathcal{B}$ has only access to the oracle O and L) cannot possibly make too many hit, even though the emulator gets $|\mathcal{C}_n|/2^{O(n)}$ bits of leakage about the oracle. Let $\mathsf{Hit}_{\mathcal{B}}^{\mathsf{O}}(L, \vec{x}, \mathcal{C}_n)$ be the random variable that counts the number of hits on $\vec{x}$ made by $\mathcal{A}$ on input $(L, \vec{x}, \mathcal{C}_n)$.

Lemma 18 (Hitting Lemma with Advice, Informal). *For every $\ell(\cdot)$, positive integers q, large enough n, challenge $\vec{x}$, L with $|W^{\overline{\mathsf{Hit}}}| = q$ and list I represented by a string length $|I| = |\mathcal{C}_n|/2^{O(n)}$, adversaries $\mathcal{C}_n, \mathcal{B}$, and for every integer $c \geq 1$,*

$$\Pr_{(W,B,H) \leftarrow \mathcal{T}|_{L_I}} \left[\mathsf{Hit}_{\mathcal{B}}^{\mathsf{O}}(L, \vec{x}, \mathcal{C}_n) \geq \frac{O(|\mathcal{C}_n|) + q}{2^n} + c + |I| \right] \leq \frac{1}{2^{\gamma \cdot c}},$$

where $\gamma > 1$, and where the probability is taken over the random sampling of $(W, B, H) \leftarrow\$ \mathcal{T}$, conditioned on L.

We first explain how the hitting lemma implies Eq. 1. First, if $\mathsf{Emu}^{\mathsf{O}}$ got a total number of hits t on $\vec{x}$, either through queries to O or through the hits contained in W^{Hit}, then conditioned on all observation seen by Emu, $\ell(n)-t$ bits of $\mathcal{L}^{\mathsf{O}}(\vec{x})$ are truly undetermined. Hence,

$$\Pr_{\vec{x}\leftarrow\$\{0,1\}^{\ell\cdot n},(\mathsf{O},\mathsf{Inv})\leftarrow\$\mathcal{T}}\left[\mathsf{Emu}^{\mathsf{O}}(L_{\bar{I}},\vec{x},\mathcal{C}_n)=\mathcal{L}^{\mathsf{O}}(\vec{x})\mid \mathsf{Emu}\text{ gets}\leq t\text{ hits on }\vec{x}\right]\leq 2^{-(\ell-t)}.$$

Now, the number of hits seen by Emu is bounded by $\mathsf{Hit}^{\mathsf{O}}_{\mathsf{Emu}}(L_{\bar{I}},\vec{x},\mathcal{C}_n)+|W^{\mathsf{Hit}}|$, where $|W^{\mathsf{Hit}}|$ is at most $\mathsf{poly}(n)\cdot\frac{\tilde{O}(|\mathcal{C}_n|)}{2^n}$: this follows from the fact that the number of hits in W^{Hit} is bounded by design by the fact that Inv on input (f,y) only returns light paths, which cannot contain more than $\mathsf{poly}(n)\cdot\frac{\tilde{O}(|f|)}{2^n}$ hits. The result follows by relying on the fact that

$$\begin{aligned}
&\Pr_{\vec{x}\leftarrow\$\{0,1\}^{\ell\cdot n},(\mathsf{O},\mathsf{Inv})\leftarrow\$\mathcal{T}}\left[\mathsf{Emu}^{\mathsf{O}}(L_{\bar{I}},\vec{x},\mathcal{C}_n)=\mathcal{L}^{\mathsf{O}}(\vec{x})\right]\\
&=\sum_t \Pr[\mathsf{Emu}\text{ gets}\leq t\text{ hits on }\vec{x}]\cdot\Pr\left[\mathsf{Emu}^{\mathsf{O}}(L_{\bar{I}},\vec{x},\mathcal{C}_n)=\mathcal{L}^{\mathsf{O}}(\vec{x})\mid \mathsf{Emu}\text{ gets }t\text{ hits}\right]\\
&\leq\sum_t 2^{-(\ell-t)}\cdot\Pr[\mathsf{Emu}\text{ gets}\leq t\text{ hits on }\vec{x}].
\end{aligned}$$

Now, the bound of Eq. 1 will be obtained by plugging the bound on

$$\Pr[\mathsf{Emu}\text{ gets }\ \leq t\text{ hits on }\vec{x}]\leq\mathsf{Hit}^{\mathsf{O}}_{\mathsf{Emu}}(L,\vec{x},\mathcal{C}_n)+|W^{\mathsf{Hit}}|,$$

by using the hitting lemma to bound $\mathsf{Hit}^{\mathsf{O}}_{\mathsf{Emu}}(L,\vec{x},\mathcal{C}_n)$. The proof then follows from the hitting lemma, to which we devote Sect. 6.

5.3 Proving Theorem 17

Let $f:\{0,1\}^*\to\{0,1\}^*$ be an oracle function. We exhibit an efficient inverter $\mathcal{A}^{\mathsf{Inv}}(f,.)$ for f, such that

$$\Pr_{z\leftarrow\$\{0,1\}^m,(\mathsf{O},\mathsf{Inv})\leftarrow\$\mathcal{T}}\left[f^{\mathsf{O}}(\mathcal{A}^{\mathsf{O},\mathsf{Inv}}(f,f^{\mathsf{O}}(z)))=f^{\mathsf{O}}(z)\right]\approx 1.$$

$\mathcal{A}$ works as follows: to invert a function $f:\{0,1\}^m\mapsto\{0,1\}^*$ given an image y, it queries Inv $\log^3 m$ times on independent inputs (f_k,y), where each f_k are syntactically different but functionally equivalent to f (this guarantees that the failure probabilities introduced by the choice of the hash function h are independent). Then, it takes a path p returned by any successful query to Inv (if any), and returns a uniformly random preimage z consistent with this path (this requires a single query to the PSPACE oracle). The proof that $\mathcal{A}$ is a successful inverter proceeds by a sequence of lemmas. First, we define f_{approx} as $f_s=\mathsf{shave}(f)$, except that it outputs $\perp$ on any input z such that the path in $f^{\mathsf{O}}_s(z)$ is not light.

First Lemma. The first lemma states that

$$\Pr_{\mathsf{O},z\leftarrow\$\{0,1\}^m}\left[f^{\mathsf{O}}_{\mathsf{approx}}(z)=f^{\mathsf{O}}_s(z)\right]\approx 1.$$

This lemma will follow again from the Hitting lemma, which provides a strong concentration bound on the probability that the path of $f_s^{\mathsf{O}}(z)$ is light: by this concentration bound, it follows that the path is light with probability at least $1 - \log|f| \cdot 2^{-O(\log^2|f|)}$ (recall that a path is heavy if, for some k, it contains $N(k)$ k-Chk queries, and more than $O(N(k)) + \log^2|f|$ hits).

Second Lemma. The second lemma states that

$$\Pr_{\mathsf{O},z\leftarrow\$\{0,1\}^m}\left[f_s^{\mathsf{O}}(z) = f^{\mathsf{O}}(z)\right] \approx 1.$$

This lemma follows from the definition of shaving: since only Chk gates with $k \geq 6\log|f|$ are shaved, the probability that $f_s^{\mathsf{O}}(z) \neq f^{\mathsf{O}}(z)$ is bounded by the sum $\sum_{k\geq 6\log(|f|)} 2^{-\frac{5k}{6}} \leq 4/m^3$. Combining the above lemmas with an averaging argument, we will show that

$$\Pr_{z\leftarrow\$\{0,1\}^m}\left[f(f^{-1}_{\mathsf{approx}}(f(z), 1^m)) = f(z)\right] \approx 1.$$

When $\mathcal{A}$ makes a single query to Inv, its overall success probability is approximately $1/8$. Since all queries have independent probability of failing due to an unfortunate choice of h, we will show that $\mathcal{A}$ inverts successfully with probability

$$\Pr_{z\leftarrow\$\{0,1\}^m,(\mathsf{O},\mathsf{Inv})\leftarrow\$\mathcal{T}}\left[f^{\mathsf{O}}(\mathcal{A}^{\mathsf{O},\mathsf{Inv}}(f, f^{\mathsf{O}}(x))) = f^{\mathsf{O}}(x)\right] \approx 1 - \left(\frac{7}{8}\right)^{\log^3 m}.$$

Note that $\mathcal{A}$, on input f, sends $\log^3 m \leq \log^3|f|$ queries to Inv, selects one of the path from the successful queries, and queries it to the PSPACE oracle to select the preimage z it outputs. Therefore, the size of $\mathcal{A}$ is $|\mathcal{A}| = \tilde{O}(|f|)$.

6 The Hitting Lemma

For any $\vec{r} = r_1 \cdots r_\ell$, we define an oracle $\mathsf{Guess}_{\vec{r}}(i, r^*)$ as taking an input r^* and an index i and checking whether $r_i = r^*$. If so, the oracle returns 1. Else, the oracle returns $\perp$. We define $\mathsf{Hit}^{\mathsf{Guess}_{\vec{r}}}(\mathcal{A})$ as the number of distinct queries $\mathcal{A}$ makes which returns something different than $\perp$.

Lemma 19 (Abstract Hitting Lemma). *For every positive integer q, large enough n, $\ell = \ell(n)$, sets $V_1, \cdots, V_\ell$ of size $1 \leq |V_i| \leq 2^n$ such that $q = \ell \cdot 2^n - \sum_{i=1}^{\ell}|V_i|$, for every adversary $\mathcal{A}$, and for every integer $c \geq 1$, $\exists\alpha > 0$, $\exists\gamma > 1$:*

$$\Pr_{\vec{r}\leftarrow\$V_1\times\cdots\times V_\ell}\left[\mathsf{Hit}^{\mathsf{Guess}_{\vec{r}}}(\mathcal{A}) \geq \frac{16\cdot \mathsf{qry}_{\mathcal{A}} + q}{2^n} + c\right] \leq \frac{\alpha}{2^{\gamma c}}.$$

The hitting lemma gives a strong Chernoff-style bound on the number of distinct hits which an arbitrary adversary $\mathcal{A}$ can make using $\mathsf{qry}_{\mathcal{A}}$ queries. The strength of this bound allows to show that the bound degrades gracefully even if $\mathcal{A}$ is additionally given an arbitrary *advice string* of bounded size about the truth table of the Guess oracle. We discuss applications and variants of the Hitting Lemma in the full version of this work [BC20], and now turn to its proof.

6.1 Proof of the Hitting Lemma – Proof Structure

The goal of $\mathcal{A}$ is to find as many distinct r_i's as possible, where each r_i is sampled randomly from a set V_i of size $|V_i| \leq 2^n$, given access to an oracle which indicates whether a guess is correct or not. Intuitively, $\mathcal{A}$'s best possible strategy is to first choose the smallest set V_{i_1}, query its elements to Guess (in arbitrary order) until it finds r_{i_1}, then move on to the second smallest set V_{i_2}, and so on. The proof of the abstract hitting lemma closely follows this intuition: we first show that this strategy is indeed the best possible strategy, then bound it's success probability using a second moment concentration bound. Formally, for any $Q \geq 1$, let $\mathcal{B}_Q$ be a Q-query adversary that implements the following simple strategy: order $V_1, \cdots, V_\ell$ by increasing size, as $V_{\sigma(1)}, \cdots, V_{\sigma(\ell)}$ for some fixed permutation σ such that $|V_{\sigma(1)}| \leq \cdots \leq |V_{\sigma(\ell)}|$. For every $i \leq \ell$, let $v_i \leftarrow |V_{\sigma(i)}|$, and let f_i be an arbitrary bijection between $[v_i]$ and $V_{\sigma(i)}$. The algorithm $\mathcal{B}_Q$ is given on Fig. 3.

Algorithm $\mathcal{B}_Q$

$\mathsf{qry} \leftarrow 0$; $r_1^*, \cdots, r_\ell^* \leftarrow \perp$
for $i = 1$ **to** ℓ :
  for $j \in [1, v_i]$:
    $\mathsf{qry} \leftarrow \mathsf{qry} + 1$
    if $\mathsf{qry} = Q$ **then return** $(r_1^*, \cdots, r_\ell^*)$
    if $\mathsf{Guess}_{\vec{r}}(i, f_i(j))$ **then** $r^*_{\sigma(i)} \leftarrow f_i(j)$; break
return $(r_1^*, \cdots, r_\ell^*)$

Fig. 3. Q-query adversary $\mathcal{B}_Q$

The adversary $\mathcal{B}_Q$ sequentially queries the values of the sets V_i ordered by increasing size, following an arbitrary ordering of the values inside each V_i, until it finds r_i (after which it moves to the next smallest larger set) or exhausts its budget of Q queries. To simplify notations, for any vector $\vec{u} \in [v_1] \times \cdots \times [v_\ell]$, we write $\pi(\vec{u}) = f_1^{-1}(u_{\sigma^{-1}(1)}), \cdots, f_\ell^{-1}(u_{\sigma^{-1}(\ell)})$. Observe that for any $t \in \mathbb{N}$,

$$\begin{aligned}\Pr_{\vec{r} \leftarrow \$ V_1 \times \cdots \times V_\ell}\left[\mathsf{Hit}^{\mathsf{Guess}_{\vec{r}}}(\mathcal{B}_Q) \geq t\right] &= \Pr_{\vec{u} \leftarrow \$ [v_1] \times \cdots \times [v_\ell]}\left[\mathsf{Hit}^{\mathsf{Guess}_{\pi(\vec{u})}}(\mathcal{B}_Q) \geq t\right] \\ &= \Pr_{\vec{u} \leftarrow \$ [v_1] \times \cdots \times [v_\ell]}\left[\sum_{i=1}^{t} u_i \leq Q\right],\end{aligned}$$

where the last equality follows from the fact that $\mathcal{B}_Q$ queries the positions one by one in a fixed order, and needs exactly u_i queries to find $r_{\sigma(i)} = f_{\sigma(i)}(u_i)$ for $i = 1$ to t. The proof of the hitting lemma derives directly from two claims. The first claim states that no Q-query adversary can make t distinct hits with probably better than that of $\mathcal{B}_Q$:

Claim 2 ($\mathcal{B}_Q$'s strategy is the best possible strategy). *For every integers n, Q, $\ell = \ell(n)$, sets $V_1, .., V_\ell$ of size $1 \leq |V_i| \leq 2^n$, and for any Q-query algorithm $\mathcal{A}$ and integer t,*

$$\Pr_{\vec{r} \leftarrow \$ V_1 \times \cdots \times V_\ell}\left[\mathsf{Hit}^{\mathsf{Guess}_{\vec{r}}}(\mathcal{A}) \geq t\right] \leq \Pr_{\vec{u} \leftarrow \$ [v_1] \times \cdots \times [v_\ell]}\left[\sum_{i=1}^{t} u_i \leq Q\right].$$

By construction, the average number of hits $\mathbb{E}_{\vec{r}}[\mathsf{Hit}^{\mathsf{Guess}_{\vec{r}}}(\mathcal{B}_Q)]$ made by $\mathcal{B}_Q$ is the largest value m such that $\sum_{i=1}^{m} \frac{v_i+1}{2} \leq Q$. Recall that $q = \ell \cdot 2^n - \sum_{i=1}^{\ell} |V_i| = \ell \cdot 2^n - \sum_{i=1}^{\ell} v_i$ and $v_i \leq 2^n$ for every i, which implies in particular that $\sum_{i=1}^{m} v_i \geq m \cdot 2^n - q$. We thus bound m as a function of Q, q, and 2^n:

$$\sum_{i=1}^{m} \frac{v_i+1}{2} \leq Q \iff m + \sum_{i=1}^{m} v_i \leq 2Q$$
$$\implies m + m \cdot 2^n - q \leq 2Q \iff m \leq \frac{2Q+q}{2^n+1}.$$

The second claim states, in essence, that the probability over $\vec{r}$ that $\mathcal{B}_Q$ does t hits decreases exponentially with the distance of t to the mean m (up to some multiplicative constant).

Claim 3 (Bounding $\mathcal{B}_Q$'s number of hits). *There exists constants $\alpha > 0$ and $\gamma > 1$ such that for every $\ell(\cdot)$, positive integers q, Q, large enough n, integers $v_1, \cdots, v_\ell$ with $1 \leq v_i \leq 2^n$ such that $q = \ell \cdot 2^n - \sum_{i=1}^{\ell} v_i$, and for every integer $c \geq 1$,*

$$\Pr_{\vec{u} \leftarrow \$[v_1] \times \cdots \times [v_\ell]} \left[\sum_{i=1}^{t} u_i \leq Q \right] \leq \frac{\alpha}{2^{\gamma c}}, \text{ where } t = \frac{16 \cdot Q + q}{2^n} + c.$$

We prove Claim 2 and Claim 3 in the full version of this work [BC20].

Acknowledgements. We thank Félix Richart for help with the experimental verification of some probability claims, and the anonymous Eurocrypt reviewers for their careful proofreading of the paper. C. Brzuska supported by the academy of Finland. G. Couteau supported by the ANR SCENE.

References

[AGGM06] Akavia, A., Goldreich, O., Goldwasser, S., Moshkovitz, D.: On basing one-way functions on NP-hardness. In: 38th ACM STOC, pp. 701–710. ACM Press, May 2006

[BB15] Bogdanov, A., Brzuska, C.: On basing size-verifiable one-way functions on NP-hardness. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part I. LNCS, vol. 9014, pp. 1–6. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46494-6_1

[BBF13] Baecher, P., Brzuska, C., Fischlin, M.: Notions of black-box reductions, revisited. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part I. LNCS, vol. 8269, pp. 296–315. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-42033-7_16

[BC20] Brzuska, C., Couteau, G.: Towards fine-grained one-way functions from strong average-case hardness. Cryptology ePrint Archive, Report 2020/1326 (2020). https://eprint.iacr.org/2020/1326

[BGI08] Biham, E., Goren, Y.J., Ishai, Y.: Basing weak public-key cryptography on strong one-way functions. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 55–72. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78524-8_4

[BHK+11] Brassard, G., Høyer, P., Kalach, K., Kaplan, M., Laplante, S., Salvail, L.: Merkle puzzles in a quantum world. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 391–410. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22792-9_22

[BM09] Barak, B., Mahmoody-Ghidary, M.: Merkle puzzles are optimal—an $O(n^2)$-query attack on any key exchange from a random oracle. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 374–390. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03356-8_22

[BR93] Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: ACM CCS 1993, pp. 62–73. ACM Press, November 1993

[BRSV17] Ball, M., Rosen, A., Sabin, M., Vasudevan, P.N.: Average-case fine-grained hardness. In: 49th ACM STOC, pp. 483–496. ACM Press, June 2017

[BRSV18] Ball, M., Rosen, A., Sabin, M., Vasudevan, P.N.: Proofs of work from worst-case assumptions. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part I. LNCS, vol. 10991, pp. 789–819. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96884-1_26

[BS08] Brassard, G., Salvail, L.: Quantum Merkle puzzles. In: Second International Conference on Quantum, Nano and Micro Technologies (ICQNM 2008), pp. 76–79. IEEE (2008)

[BT03] Bogdanov, A., Trevisan, L.: On worst-case to average-case reductions for NP problems. In: 44th FOCS, pp. 308–317. IEEE Computer Society Press, October 2003

[CDGS18] Coretti, S., Dodis, Y., Guo, S., Steinberger, J.P.: Random oracles and non-uniformity. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part I. LNCS, vol. 10820, pp. 227–258. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78381-9_9

[DGK17] Dodis, Y., Guo, S., Katz, J.: Fixing cracks in the concrete: random oracles with auxiliary input, revisited. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part II. LNCS, vol. 10211, pp. 473–495. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56614-6_16

[DVV16] Degwekar, A., Vaikuntanathan, V., Vasudevan, P.N.: Fine-grained cryptography. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part III. LNCS, vol. 9816, pp. 533–562. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53015-3_19

[FF93] Feigenbaum, J., Fortnow, L.: Random-self-reducibility of complete sets. SIAM J. Comput. **22**(5), 994–1005 (1993)

[GT00] Gennaro, R., Trevisan, L.: Lower bounds on the efficiency of generic cryptographic constructions. In: 41st FOCS, pp. 305–313. IEEE Computer Society Press, November 2000

[Hel80] Hellman, M.: A cryptanalytic time-memory trade-off. IEEE Trans. Inf. Theory **26**(4), 401–406 (1980)

[HL18] Holmgren, J., Lombardi, A.: Cryptographic hashing from strong one-way functions (or: one-way product functions and their applications). In: 59th FOCS, pp. 850–858. IEEE Computer Society Press, October 2018

[HR04] Hsiao, C.-Y., Reyzin, L.: Finding collisions on a public road, or do secure hash functions need secret coins? In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 92–105. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28628-8_6

[Imp95] Impagliazzo, R.: A personal view of average-case complexity. In: Proceedings of Structure in Complexity Theory. Tenth Annual IEEE Conference, pp. 134–147. IEEE (1995)

[IR89] Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: 21st ACM STOC, pp. 44–61. ACM Press, May 1989

[IR90] Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 8–26. Springer, New York (1990). https://doi.org/10.1007/0-387-34799-2_2

[Lev86] Levin, L.A.: Average case complete problems. SIAM J. Comput. **15**(1), 285–286 (1986)

[Lev87] Levin, L.A.: One way functions and pseudorandom generators. Combinatorica **7**(4), 357–363 (1987)

[LLW19] LaVigne, R., Lincoln, A., Williams, V.V.: Public-key cryptography in the fine-grained setting. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part III. LNCS, vol. 11694, pp. 605–635. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26954-8_20

[Mer78] Merkle, R.C.: Secure communications over insecure channels. Commun. ACM **21**(4), 294–299 (1978)

[PL20] Pass, R., Liu, Y.: On one-way functions and Kolmogorov complexity. In: FOCS 2020 (2020)

[PV20] Pass, R., Venkitasubramaniam, M.: Is it easier to prove statements that are guaranteed to be true? In: FOCS 2020 (2020)

[RTV04] Reingold, O., Trevisan, L., Vadhan, S.P.: Notions of reducibility between cryptographic primitives. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 1–20. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24638-1_1

[Sim98] Simon, D.R.: Finding collisions on a one-way street: can secure hash functions be based on general assumptions? In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 334–345. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0054137

[STV01] Sudan, M., Trevisan, L., Vadhan, S.: Pseudorandom generators without the XOR lemma. J. Comput. Syst. Sci. **62**(2), 236–266 (2001)

[Sud97] Sudan, M.: Decoding of Reed Solomon codes beyond the error-correction bound. J. Complex. **13**(1), 180–193 (1997)

[Unr07] Unruh, D.: Random oracles and auxiliary input. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 205–223. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74143-5_12

[VV85] Valiant, L.G., Vazirani, V.V.: NP is as easy as detecting unique solutions. In: 17th ACM STOC, pp. 458–463. ACM Press, May 1985

[WB86] Welch, L.R., Berlekamp, E.R.: Error correction for algebraic block codes (1986). US Patent 4,633,470

[Wee06] Wee, H.: Finding Pessiland. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 429–442. Springer, Heidelberg (2006). https://doi.org/10.1007/11681878_22

[Yao82] Yao, A.C.-C.: Theory and applications of trapdoor functions (extended abstract). In: 23rd FOCS, pp. 80–91. IEEE Computer Society Press, November 1982