On the Compressibility of $\mathcal{NP}$ Instances and Cryptographic Applications[‡]

Danny Harnik[*] Moni Naor[†]

Abstract

We study compression that preserves the *solution* to an instance of a problem rather than preserving the instance itself. Our focus is on the compressibility of $\mathcal{NP}$ decision problems. We consider $\mathcal{NP}$ problems that have long instances but relatively short witnesses. The question is, can one efficiently compress an instance and store a shorter representation that maintains the information of whether the original input is in the language or not. We want the length of the compressed instance to be polynomial in the length of the *witness* and polylog in the length of original input. We discuss the differences between this notion and similar notions from parameterized complexity. Such compression enables to succinctly store instances until a future setting will allow solving them, either via a technological or algorithmic breakthrough or simply until enough time has elapsed.

We give a new classification of $\mathcal{NP}$ with respect to compression. This classification forms a stratification of $\mathcal{NP}$ that we call the $\mathcal{VC}$ hierarchy. The hierarchy is based on a new type of reduction called W-reduction and there are compression-complete problems for each class.

Our motivation for studying this issue stems from the vast cryptographic implications compressibility has. For example, we say that SAT is compressible if there exists a polynomial $p(\cdot,\cdot)$ so that given a formula consisting of m clauses over n variables it is possible to come up with an equivalent (w.r.t satisfiability) formula of size at most $p(n, \log m)$. Then given a compression algorithm for SAT we provide a construction of collision resistant hash functions from *any* one-way function. This task was shown to be impossible via black-box reductions [71], and indeed the construction presented is inherently non-black-box. Another application of SAT compressibility is a cryptanalytic result concerning the limitation of everlasting security in the bounded storage model when mixed with (time) complexity based cryptography. In addition, we study an approach to constructing an Oblivious Transfer Protocol from *any* one-way function. This approach is based on compression for SAT that also has a property that we call *witness retrievability*. However, we mange to prove severe limitations on the ability to achieve witness retrievable compression of SAT.

1 Introduction

In order to deal with difficult computational problems several well established options were developed, including: approximation algorithms, subexponential algorithms, parametric complexity and average-case complexity. In this paper we explore our favorite approach for dealing with problems: *postpone* them (hopefully, without cluttering our desk or disk). We initiate the study of the compressibility of NP problems for

[*]Department of Computer Science, Technion, Haifa, Israel. E-mail: `harnik@cs.technion.ac.il`. This research was conducted while at the Weizmann Institute, Rehovot.

[†]Incumbent of the Judith Kleeman Professorial Chair, Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot 76100 Israel. Email: `moni.naor@weizmann.ac.il`.

[‡]Research supported by a grant from the Israel Science Foundation. A short version of this paper appeared in FOCS 2006.

their resolution in some future setting and in particular the cryptographic significance of such compression. Rather than solving a given instance, we ask whether a shorter instance with the same solution can be found efficiently. We emphasize that we are not interested in maintaining the information about the original instance (as is the case in typical notions of compression), but rather maintain the solution only. The solution can possibly be much shorter than the input (as short as a yes/no answer), thus the potential of such a compression is high.

Specifically, we consider $\mathcal{NP}$ problems that have long instances but relatively short witnesses. An $\mathcal{NP}$ language L is defined by an efficiently computable relation R_L such that an input (or instance) x is in L if and only if there exists a witness w such that $R_L(x, w) = 1$. Throughout the paper, an $\mathcal{NP}$ instance is characterized by two parameters m and n: The length of the instance x is denoted by m and the length of the witness w is denoted by n. The problems of interest are those having short witnesses, i.e. $n << m$. Traditionally, the study of $\mathcal{NP}$ languages evolves around the ability or inability to efficiently decide if an instance is in the language or not, or to find a witness w for an instance x within polynomial time. We introduce the question of compressibility of such instances.

Compressing SAT Instances: To illustrate the relevant setting, we use the well known example of SAT. An instance Φ for SAT consists of a CNF formula over n variables and we define that $\Phi \in SAT$ if there exists an assignment to the n variables that satisfies all the clauses of Φ. We begin with compressibility with respect to decision, and discuss compressibility of the search variant later in the paper. The question of compressibility of SAT is the following:

Example 1.1 (Compression of SAT instances)
Does there exist an efficient algorithm and a polynomial $p(\cdot, \cdot)$ with the following input and output:
Input: *A CNF formula Φ with m clauses over n variables (we are interested in $m >> n$).*
Output: *A formula Ψ of size $p(n, \log m)$ such that Ψ is satisfiable if and only if Φ is satisfiable.*

The idea is that the length of Ψ should be essentially unrelated to the original length m, but rather to the number of variables (or in other words, to the size of the witness). Typically, we think of the parameters m and n as related by some function, and it is instructive (but not essential) to think of m as larger than any polynomial in n. So potentially, the length of Ψ can be significantly shorter than that of Φ.[1]

In general, one cannot expect to compress all the formulas, or else we would have an efficient algorithm for all $\mathcal{NP}$ problems.[2] However, once we introduce the setting of a shorter witness, then compression becomes plausible. Note that if $\mathcal{P} = \mathcal{NP}$ and we actually know the algorithm for SAT, then clearly compression is trivial, simply by solving the satisfiability of Φ and outputting 1 if $\Phi \in SAT$ and 0 otherwise.

Motivation for Compression: Compressing for the future is an appealing notion for various settings. There are numerous plausible scenarios that will give us more power to solve problems in the future. We could potentially find out that $\mathcal{P} = \mathcal{NP}$ and solve all our $\mathcal{NP}$ problems then. We may have faster computers or better means of computing such as quantum computers or any other physical method for solving problems (see Aaronson [1] for a list of suggestions). Above all, the future entails lots and lots of time, a resource that the present is usually short of. Saving the problems of today as they are presented is wasteful, and compression of problems will allow us to store a far greater number of problems for better days.

[1] Note, that since our requirement for compression is only relevant for problems where $m >> n$, then an $\mathcal{NP}$-complete problem such as 3-SAT (where all clauses have exactly 3 literals) is irrelevant for compression as in such formulas m is already at most $O(n^3)$.

[2] Suppose that every formula can be compressed by a single bit, then sequentially reapplying compression to the input will result in a very short formula that may be solved by brute enumeration.

Our interest in studying the issue of compression stems from the vast cryptographic implications of compressibility. We demonstrate three questions in cryptography that compression algorithms would resolve (see Section 1.3). We are confident that the compression of problems implies further applications both within and outside of cryptography. For example, in subsequent works, Dubrov and Ishai [26] show the relevance of the notion of compression to derandomization and Dziembowski [28] shows that compression is related to the study of forward-secure storage (see Section 1.4 on related work). We note that a notion similar to compression has been useful (and well studied) in the context of parameterized complexity (see a comparison and discussion in Section 1.4). The concept of compression of problems is also interesting beyond the confines of $\mathcal{NP}$ problems, and makes sense in any setting where the compression requires much less resources than the actual solution of the problem.

1.1 Compression of NP instances

We define the notion of compression with respect to an $\mathcal{NP}$ language. For simplicity, we assume that an input to an $\mathcal{NP}$ language L includes an encoding of the parameter n that upper bounds the length of a potential witness.[3] We also associate with L a specific $\mathcal{NP}$ relation R_L that defines it (as mentioned above). We note that once the parameters m and n are explicit, it is in most cases immaterial what specific relation defines the language and the properties we discuss (such as compressibility) are properties of the language at hand (unless stated otherwise). In essence, a compression algorithm is a specialized Karp-reduction that also reduces the length of the instance.

Definition 1.2 (Compression Algorithm for $\mathcal{NP}$ Instances) *Let L be an $\mathcal{NP}$ language where m and n denote the instance length and the witness length respectively. A* compression algorithm *for L is a probabilistic polynomial time machine Z along with a language L' in $\mathcal{NP}$ (or more accurately in $\mathcal{NP}(poly(m))$)*[4] *and a polynomial $p(\cdot,\cdot)$ such that for all large enough m:*

1. *For all $x \in \{0,1\}^m$ with parameter n the length of $Z(x)$ is at most $p(n, \log m)$.*
2. *$Z(x) \in L'$ if and only if $x \in L$*

The above definition is of an errorless *compression. We also consider a probabilistic variant called ε-*compression *for some real function $\varepsilon : \mathbb{N} \to [0,1]$. The probabilistic definition is identical to the errorless one except for the second property that is augmented to:*

2'. *For large enough n, for all $x \in \{0,1\}^m$ with parameter n it holds that:*

$$\Pr[(Z(x) \in L') \Leftrightarrow (x \in L)] > 1 - \varepsilon(n)$$

where probability is over the internal randomness of Z.Typically we require $\varepsilon(\cdot)$ to be negligible.

The paper consists of two parts: *Part I* is a study of the concept of compression of $\mathcal{NP}$ instances from a complexity point of view. *Part II* introduces the cryptographic applications of compression algorithms.

[3] Typically, the parameter n is indeed part of the description of the problem (e.g. for Clique, SAT, Long-path and others).

[4] By $\mathcal{NP}(poly(m))$ we mean in nondeterministic-time $poly(m)$ (that is, verifiable in time $poly(m)$ when given a non-deterministic hint).

How much to compress: Definition 1.2 (of compression algorithms) requires a very strong compression, asking that the length of the compression is polynomial in n and $\log m$. For the purposes of part I of the paper (the complexity study), it is essential that the compression is at least sub-polynomial in m in order to ensure that the reductions defined with respect to compressibility (See Section 2.2) do compose.[5] Furthermore, for part II (the applications) this definition may be strongly relaxed, where even a compression to $m^{1-\varepsilon}$ for some constant ε suffices for some applications.

The Complexity of L': Another requirement of Definition 1.2 is that the language L' be in $\mathcal{NP}(poly(m))$. In general, this requirement may also be relaxed and the result still be meaningful for some applications. In particular, we do not need to put a bound on the complexity of L', but only require that there is enough information in $Z(x)$ to determine whether $x \in L$ or not. One case where we use a definition with unbounded extraction is the compression of search problems in Section 2.7. It should be noted however that in some settings the requirement for L' to be in $\mathcal{NP}(poly(m))$ is essential, such as when defining the witness retrievability property (Definition 1.5). Moreover, in some cases it is natural to further restrict L' to actually be in $\mathcal{NP}$ (that is in $\mathcal{NP}(poly(n, \log m))$. For instance, this is the case in the definition of compression of SAT (Example 1.1). Finally, note that if the compression is *errorless*, then L' must be in $\mathcal{NP}(poly(m))$ simply by the definition of compression.[6]

Paper organization: In the rest of the introduction we survey the results of this paper, including part I (the complexity study) and part II (the cryptographic applications). In section 1.4 we discuss related and subsequent works. The main complexity study of the compressibility of $\mathcal{NP}$ problems appears in Section 2. The Cryptographic applications are in Sections 3,5 and 4. In Section 3 we describe the application of compression to constructing CRH from any one-way function. Section 4 presents the implication to the hybrid bounded storage model, while Section 5 discusses witness retrievable compression and its application to the construction of OT from any one-way function. We conclude with a discussion and some open problems (Section 6).

1.2 Part I: Classifying $\mathcal{NP}$ Problems with Respect to Compression.

We are interested in figuring out which $\mathcal{NP}$ languages are compressible and, in particular, whether important languages such as SAT and Clique are compressible. For starters, we demonstrate some non-trivial languages that do admit compression (Section 2.1): we show compression for the well known $\mathcal{NP}$-complete problem of vertex-cover and for another $\mathcal{NP}$-complete language known as minimum-fill-in. We show a generic compression of sparse languages (languages containing relatively few words from all possible instances). As a specific example we mention the language consisting of strings that are the output of a cryptographic pseudorandom generator and also consider the sparse subset sum problem. In addition we show compression for the promise problem GapSAT.[7] However, these examples are limited and do not shed light on the general compression of other $\mathcal{NP}$ problems. Moreover, it becomes clear that the traditional notions of reductions and completeness in $\mathcal{NP}$ do not apply for the case of compression (i.e., the compression

[5] For clarity we choose a polynomial in $\log m$, although this may be replaced by any sub-polynomial function $m'(.)$ (a function such that for large enough m for any polynomial $q(\cdot)$ we have $m'(m) < q(m)$).

[6] Suppose that there exists a compression algorithm Z for L then define L' to be the language of all $Z(x)$ such that $x \in L$. Then, for every $y \in L'$ a verification algorithm takes as a nondeterministic witness a value x, a witness to $x \in L$ along with randomness for the compression algorithm and verifies that indeed $y = Z(x)$. Thus if Z never introduces an error then L' is in $\mathcal{NP}(poly(m))$.

[7] I.e. a promise problem were either the formula is satisfiable or every assignment does not satisfy a relatively large number of clauses.

of an $\mathcal{NP}$-complete language does not immediately imply compression for all of $\mathcal{NP}$). This is not surprising since this is also the case with other approaches for dealing with $\mathcal{NP}$-hardness such as approximation algorithms or subexponential algorithms (see for example [68]) and parameterized complexity (see [24] and further discussion in Section 1.4 on related work). For each of these approaches, appropriate new reductions where developed, none of which is directly relevant to our notion of compression.

We introduce W-reductions in order to study the possibility of compressing various problems in $\mathcal{NP}$. These are reductions that address the length of the witness in addition to membership in an $\mathcal{NP}$ language. W-reductions have the desired property that if L W-reduces to L', then any compression algorithm for L' yields a compression algorithm for L. Following the definition of W-reductions we define also the matching notion of compression-complete and compression-hard languages for a class.

The $\mathcal{VC}$ classification: We introduce a classification of $\mathcal{NP}$ problems with respect to compression. The classification presents a structured hierarchy of $\mathcal{NP}$ problems, that is surprisingly different from the traditional view and closer in nature to the W hierarchy of parameterized complexity (see [24] and [33]). We call our hierarchy $\mathcal{VC}$, short for "verification classes", since the classification is closely related to the verification algorithm of $\mathcal{NP}$ languages when allowed a preprocessing stage. We give here a very loose description of the classes, just in order to convey the flavor of the classification. Formal definitions appear in Section 2.3. In the following definition, when we use the term "verification" we actually mean "verification with preprocessing":

- For $k \geq 2$, the class $\mathcal{VC}_k$ is the class of languages that have verification which can be presented as a depth k polynomial size circuit (polynomial in n and m). For example, the language SAT is compression-complete for the class $\mathcal{VC}_2$. Other examples include Integer-Programming that resides in $\mathcal{VC}_{\log n}$ and Dominating-Set that is in $\mathcal{VC}_3$. Both of which are shown to be compression-hard for $\mathcal{VC}_2$.
- $\mathcal{VC}_1$ is the class of languages that have *local* verification. That is, languages which can be verified by testing only a small part (of size $poly(n, \log m)$) of the instance. This class contains many natural examples such as the Clique language or Long-path.
- $\mathcal{VC}_{OR}$ is the class of languages that have verification which can be presented as the OR of m small instances of SAT (each of size n). This class contains the languages that are relevant for the cryptographic applications. The Clique language is compression-hard for this class (Claim 2.22).
- $\mathcal{VC}_0$ is the class of compressible languages. In particular it includes vertex cover, sparse languages and GapSAT.

We show that the classes described form a hierarchy (see Lemma 2.17 and Claim 2.23). That is:

$$\mathcal{VC}_0 \subseteq \mathcal{VC}_{OR} \subseteq \mathcal{VC}_1 \subseteq \mathcal{VC}_2 \subseteq \mathcal{VC}_3 \ldots$$

We discuss some of the more interesting classes in the $\mathcal{VC}$ hierarchy, classify some central $\mathcal{NP}$ problems and mention compression-complete problems for the classes. Note that the existence of a compression algorithm for a complete problem for some class entails the collapse of the hierarchy up to that class into $\mathcal{VC}_0$.

In addition, we study the compression of $\mathcal{NP}$ *search* problems. That is, compressing an instance in a way that maintains all the information about a witness for the problem. We show that the compression of a class of decision problems also implies compression for the corresponding search problems. Formally:

Theorem 1.3 *If a class $\mathcal{VC}_k$ has a compression algorithm, then there is a compression algorithm for the search problem of a relation R_L of $L \in \mathcal{VC}_k$.*

This theorem turns out to be useful for the cryptanalysis result regarding the bounded storage model we present in Section 4.

1.3 Part II: Implications to Cryptography

As the main motivation for the study of compression, we provide some strong implications of compressibility to cryptography. The implications described are of contrasting flavors. On the one hand we show constructions of cryptographic primitives using compression algorithms, while on the other hand we show a cryptanalysis using compression algorithms (or alternatively, this can be considered as an application of incompressibility of languages). For simplicity we provide the implication with respect to the compression of SAT. We note however, that the same statements can actually be made with compression of languages from the class $\mathcal{VC}_{OR}$ (see Definition 2.20). This class is the lowest class in our $\mathcal{VC}$ hierarchy, and potentially easier to compress than SAT. Moreover, the instances that we need to compress for our applications are further limited in the sense that (i) the instances are in $\mathcal{NP}\cap$ Co-$\mathcal{NP}$ and (ii) the (positive and negative) instances have a unique witness.

Basing Collision Resistant Hash Functions on Any One-Way Function: Collision Resistant Hash functions (CRH) are important cryptographic primitives with a wide range of applications, e.g. [63, 18, 51, 19, 58, 6]. Loosely speaking, a CRH is a family $\mathcal{H}$ of length reducing functions, such that no efficient algorithm can find collisions induced by a random hash from the family. Currently there is no known construction of CRH from general one-way functions or one-way permutations, and moreover, Simon [71] showed that basing CRH on one-way permutations cannot be achieved using black-box reductions. We show how a general compression algorithm may be used to bridge this gap.

Theorem 1.4 *If there exists an errorless*[8] *compression algorithm for SAT then there exists a construction of collision resistant hash functions based on any one-way function.*

The construction of the CRH in Theorem 1.4 is inherently non-black-box and uses the program of the one-way function via Cook's Theorem [17]. This is essential to the validity of this approach, in light of the black-box impossibility result [71].

An interesting corollary of this result is a construction of statistically hiding bit commitment from any one-way function, which is currently an open problem. Moreover, the construction would require only a single round of interaction ([62, 38] show constructions of statistically hiding bit commitment based on one-way functions with a specific structure and also require a large number of rounds of interaction).

On Everlasting Security and the Hybrid Bounded Storage Model: The *bounded storage model* (BSM) of Maurer [55] provides the setting for the appealing notion of *everlasting security* [4, 22]. Loosely speaking, two parties, Alice and Bob, that share a secret key in advance, may use the BSM to encrypt messages in a way that the messages remain secure against a computationally unbounded adversary, even if the shared secret key is eventually revealed.

However, if the parties do not meet in advance to agree on a secret key then everlasting security requires high storage requirements from Alice and Bob [29], rendering encryption in this model less appealing. Hoping to overcome this, it was suggested to combine the BSM with computational assumptions (what is called here the hybrid BSM). In particular, to run a computational key agreement protocol in order to agree on a shared secret key, and then run one of the existing BSM schemes. Dziembowski and Maurer [29]

[8]The construction of CRH requires that the error probability of compression algorithm will be zero. This can be slightly relaxed to an error that is exponentially small in m (rather than n).

showed that this idea does not necessarily work in all cases, by showing an attack on a protocol consisting of the combination of a specific (artificial) computational key agreement protocol with a specific BSM encryption scheme.

We show that compression of $\mathcal{NP}$ instances can be used to attack *all* hybrid BSM schemes. Or in other words, if a compression of SAT exists, then the hybrid BSM is no more powerful than the standard BSM. One interpretation of this result is that in order to prove everlasting security for a hybrid BSM scheme, without further conditions, one must prove that there exists no compression algorithm for SAT. Alternatively, as a relaxation, one should come up with a reasonable incompressibility assumption regarding the resulting formulae. Note however that a straightforward assumption of the form "this distribution on SAT formulae is incompressible" is not efficiently falsifiable, in the sense of Naor [61], that is, it is not clear how to set up a challenge that can be solved in case the assumption is false.

On Random Oracles: The authors of this paper show in [40] that if all parties are given access to a random oracle, then there actually exists everlasting security in the hybrid BSM without an initial key and with low storage requirements from Alice and Bob[9]. Therefore, finding a compression algorithm for SAT would show an example of a task that is simple with random oracles but altogether impossible without them. This would constitute an argument against relying (blindly) on random oracles to determine whether a task is feasible at all. This is different than previous results such as [13, 37, 57, 7] that show a specific protocol that becomes insecure if the random oracle is replaced by a function with a small representation. Model separation results (such as the implication of compression) were discussed by Nielsen [66, 67] (for non-interactive non-committing encryption) and Wee [76] (for obfuscating point functions), but the separation there is between the programmable and non-programmable random oracle models (in contrast, the hybrid BSM result in [40] holds also if the oracle is non-programmable).

The Actual Model and Results: The *bounded storage model* bounds the storage space of an adversary rather than its running time. It utilizes the public transmission of a long random string $\mathcal{R}$ of length m (sometimes referred to as the broadcast string), and relies on the assumption that an eavesdropper cannot possibly store all of this string. The everlasting security achieved by encryption schemes in this model means that an encrypted message remains secure even if the adversary eventually gains more storage or gains knowledge of (original) secret keys that may have been used. However, if the honest parties do not share any private information in advance, then achieving everlasting security requires storage capacity of $\Omega(\sqrt{m})$ from the honest parties (as shown in [29]).

The *hybrid bounded storage model* (see [40] for formal definitions and notions of security) assumes computational limitations on the eavesdropper up until the time that the transmission of $\mathcal{R}$ has ended. Computational assumptions with such a strict time limit are generally very reasonable. For instance, in the key agreement example, all that we require is that the computational protocol is not broken in the short time period between its execution and the transmission of $\mathcal{R}$. An assumption such as the Diffie-Hellman key agreement [21] cannot be broken within half an hour, can be made with far greater degree of trust than actually assuming the long term security of a computational key agreement protocol. We consider two models, and give a cryptanalysis result for each of them:

- **The Basic BSM Scheme:** The honest parties may only interact before the broadcast of $\mathcal{R}$ (except for actually sending the encrypted message). Thus the encryption key is fully determined at the end of

[9]This does not contradict the compressibility of SAT, since the cryptanalytic result is not black-box and assumes access to the full description of the programs of Alice and Bob. Thus this result is not preserved in the presence of a random oracle.

the broadcast of $\mathcal{R}$. Such a scheme is fully breakable in the standard BSM (without initial keys). We show that compression of SAT allows to break any basic hybrid scheme (Theorem 4.2).[10]

- **The General BSM Scheme:** Alice and Bob can interact both before *and* after the broadcast of $\mathcal{R}$. In the standard BSM (without initial keys) such a scheme is breakable unless Alice and Bob use storage of size $\Omega(\sqrt{m})$. In the hybrid BSM, we show (Theorem 4.4) that if a compression of SAT exists then such a scheme is breakable unless Alice and Bob use storage of size $\Omega(\sqrt{m/p(n, \log m)})$, where n is the security parameter of the computational protocol and p is a polynomial (related to the polynomial of the compression algorithm and the running time of the protocol that Alice and Bob use).

Witness retrievable compression and the existence of Minicrypt: The next application is an attempt to use compression in order to prove, in the terminology of [44], that `Minicrypt=Cryptomania`. Impagliazzo [44] summarizes five possibilities for how the world may look like based on different computational assumptions. The two top worlds are `Minicrypt`, where one-way functions exist but oblivious transfer protocols do not exist (in this world some interesting cryptographic applications are possible, and in particular *shared* key cryptography exists) and `Cryptomania` where Oblivious Transfer protocols do exist (and hence also a wide range of cryptographic applications like secure computation and *public* key cryptography). Whether OT can be constructed from any one-way function is a major open problem in cryptography. Impagliazzo and Rudich [46] addressed this problem and proved that key agreement protocols (and hence also OT) cannot be constructed from any one-way function using black-box reductions.

We explore an approach of using compression in order to bridge the gap between the two worlds. In order to do so we introduce an additional requirement of a compression algorithm.

Definition 1.5 (Witness Retrievable Compression) *Let Z, L and L' define a compression algorithm as in Definition 1.2 and let R_L be an $\mathcal{NP}$ relation for L. The compression is said to be* witness retrievable *with respect to R_L if there exists a probabilistic polynomial time machine W such that if input $x \in L$ then for every witness w_x for R_L it holds that $w_y = W(w_x, Z(x))$ is a witness for $Z(x) \in L'$. We allow a negligible error in the success of W (where probability is over the internal randomness of Z and W).*

Theorem 1.6 *If there exists a witness retrievable compression algorithm for a certain type of SAT formulas, then there exists an Oblivious Transfer protocol based on* any *one-way function.*

As in the CRH construction (Theorem 1.4), the construction of OT in Theorem 1.6 is inherently non-black-box. Unfortunately we show that this approach cannot work with a compression algorithm for the *general* SAT problem, due to the following theorem:[11]

Theorem 1.7 *If one-way functions exist then there is no witness retrievable compression of SAT.*

Furthermore, we also rule out the possibility of other types of witness retrievable compression that may be sufficient for Theorem 1.6. More precisely, the inability of witness retrievable compression does not change when allowing an error in the retrieval, or when dealing with a case where there is a unique witness (see Corollary 5.7). These developments rule out basing the approach of Theorem 1.6 on the compression of a general and standard language. The approach may still work out with a witness retrievable compression algorithm for the specific CNF formulas as stated in Theorem 1.6.

[10] Basic schemes are very relevant to the hybrid BSM as they include a combination of a key agreement protocol with a private key scheme (such as the scheme described by [29]).

[11] The first version of this paper [41] (dated Feb 17, 2006) did not contain this theorem and was hence more optimistic on the possibility of finding a witness preserving compression algorithm for SAT.

Finally, we point out that almost all of the examples of compression algorithms in this paper (in Sections 2.1 and 2.9) are in fact witness retrievable. This demonstrates that these examples fall short of the general compression that we are seeking. In fact a major obstacle in achieving compression for problems such as SAT seems to be that most ideas are witness retrievable.

1.4 Related Work

The relationship between compression and complexity in general is a topic that has been investigated since the early days of Complexity Theory (i.e. Kolmogorov Complexity [53]). However, the feature that we are studying in this work is compressibility with respect to the *solution* (witness) rather than the instance. The goal of maintaining the solution differs our work from a line of seemingly related works about notions of compression ([27, 72, 75] to name a few), all of which aim at eventually retrieving the input of the compression algorithm.

There are several examples of other relaxations to solving $\mathcal{NP}$ problems in polynomial time. Each of these relaxations yields a corresponding classifications of $\mathcal{NP}$. These classifications, like ours, are subtle and usually turn out to be different than the traditional $\mathcal{NP}$ classification. For example, Papadimitriou and Yannakakis [69] introduce L-reductions and the classes MAX NP and MAX SNP, with respect to approximation algorithms. Impagliazzo, Paturi and Zane [45] define reductions with respect to solution in sub-exponential time.

Perhaps the most relevant classification to ours is that of *parameterized complexity* (see the monographs on this subject by Downey and Fellows [24], Niedermeier [65] and Flum and Grohe [33]). Parameterized complexity studies the tractability of problems when one of the parameters is considered to be fixed or very small (this is called fixed parameter tractability (FPT)). One of the basic techniques of acquiring efficient algorithms in this context is the method of "kernelization" that *may* yield natural compression algorithms (see examples in Section 2.1). The kernelization method first shrinks the instance to a smaller instance whose size is only a function of the parameter and then solves it in brute force. However, inspite of the similarities between kernelization and compression, they are quite different notions. At a high level, kernelization is geared towards getting closer to a solution of the original instance. Our notion, on the other hand, requires compression per se, disregarding whether it is much harder to solve the compressed instance than the original one (in fact, in our main application for collision resistant hash in Section 3, the compressed instance never has to be solved). Indeed we expect that new methods of compression that would resolve the problems we raise in this paper will utilize this property (that the compressed instance is harder to solve). Due to those similarities, the *Weft* hierarchy of parameterized complexity is also reminiscent of the $\mathcal{VC}$-hierarchy: they are both defined by reductions to circuits of bounded depth. However, as discussed above, our study of compression yields quite a different classification. See a further discussion in section 2.1.

A related notion to parameterized complexity that is reminiscent of our work is *limited non-determinism*, which started with the work of Kintala and Fischer [52], see survey by Goldsmith, Levy and Mundheck [36]. This was further studied by Papadimitriou and Yannakakis [70] who defined a few syntactic classes within the class of polylog non-determinism ($LOGNP$ and $LOGSNP$). The interesting point is that several natural problems are complete for these classes. The notion of reduction used is the usual polynomial reduction and hence the classifications arising from this study are very different from our $\mathcal{VC}$ hierarchy. A related classification is the EW-hierarchy defined by Flum, Grohe and Weyer [34]. This hierarchy is a similar to the Weft classification of parameterized complexity but limits the running time to be only exponential in the witness length, thus geared towards problems with polylogarithmic size parameters (as in $LOGNP$).

Subsequent Works: Dubrov and Ishai [26] discussed the compression of problems and showed that a certain incompressibility assumption has an application to derandomization. Specifically they construct a generator that fools procedures that use more randomness than their output length. Their work was mostly conducted independently of ours, following conversations regarding an early phase of our work. In addition, inspired by our CRH construction, they prove that any one-way permutation can either be used for the above mentioned derandomization, or else can be used to construct a weak version of CRH.

In a recent paper, Dziembowski [28] shows the relevance of our notion of witness retrievable compression to a method for achieving *forward-secure storage*. He shows a cryptanalytic result of such compression. Furthermore, following our approach for construction of OT from one-way functions, he shows that for every one-way function either a specific storage scheme is forward-secure, or there exists an OT protocol based on this one-way function.

2 Part I: On the Compression of $\mathcal{NP}$ Instances

Attempting to compress $\mathcal{NP}$ instances requires a different approach than solving $\mathcal{NP}$ problems. Intuitively, a solution for compression might arise while trying to solve the problem. While a full solution of an $\mathcal{NP}$ problem may take exponential time, it is plausible that the first polynomial number of steps leaves us without an explicit solution but with a smaller instance. Indeed, some algorithms in the parameterized complexity world work like this (see some examples in the next section). On the other hand, we allow the possibility that the compressed version is actually harder to solve (computational time-wise) than the original one (and may require a somewhat longer witness altogether).

2.1 Examples of Compression Algorithms for some Hard Problems

We start by showing three examples of compression algorithms for problems that are conjectured not to be in $\mathcal{P}$. Two of these example are $\mathcal{NP}$-complete problems, while the third is taken from cryptography.

Vertex Cover: The well studied $\mathcal{NP}$-complete problem of Vertex-Cover receives as input a graph $G = (V, E)$ and asks whether there exists a subset of vertices $S \subseteq V$ of size at most k such that for every edge $(u, v) \in E$ either u or v are in S. The parameters are the instance length m, which is at most $O(|E| \log |V|)$, and the witness length $n = k \log |V|$.

Claim 2.1 *There exists a witness retrievable compression algorithm for Vertex-Cover.*

Proof: We are following the parameterized complexity algorithm for vertex-cover (presented in [24] and attributed to S. Buss). If a vertex-cover S of size k exists, then any vertex of degree greater than k must be inside the set S. The compression algorithm simply identifies all such vertices and lists them in the cover, while removing all their outgoing edges (that do not need to be covered by other vertices). The graph left after this process has maximal degree k, and furthermore all edges have at least one end in the cover. Thus, if the original graph has a k vertex cover then the total number of edges left is at most k^2 (at most k vertices in the cover with at most k edges each). If there are more then k^2 edges then the answer to the problem is NO, otherwise, such a graph can be represented by the list of all edges, which takes $k^2 \log k$ bits. The compression can be made witness retrievable since if we use the original labels of vertices to store the new graph, then the original cover is also a cover for the new compressed graph. □

It is in fact possible get the compressed instance to be a graph with $2k$ vertices, rather than k^2 edges, as shown in [15] and [16] (see [65] Chapter 7). It is interesting to note that we do not know of a compression algorithm for the Clique problem or the Dominating Set problem, which are strongly linked to the vertex-cover problem in the traditional study of $\mathcal{NP}$, and in fact, in Theorems 5.1, 3.1 and 4.2 we show strong implications of a compression algorithm for these languages.

On parameterized complexity and compression: The use of an algorithm from parameterized complexity for compression is not a coincidence. The "problem kernel" or "kernelization" method (see [24], Chapter 3 or [65] Chapter 7) is to first reduce the problem to a small sub-instance that, like compression, contains the answer to the original problem. Then the algorithm runs in time that is a function only of the sub-instance, e.g. exponential in this small instance size. As was discussed in Section 1.4, if the running time and output size of the first reduction happens to be only polynomial in the parameter (a class formally defined in [9]), then the first phase of the algorithm is a compression algorithm. Downey, Fellows and Stege [25] (Lemma 4.7) show that kernelization (with arbitrary functions of the witness) captures precisely fixed parameters problems.

In this context, it is important to note that a compression algorithm for a language *does not* necessarily give a parameterized complexity algorithm for the same language. At first glance it seems that one can first run the compression algorithm, and then solve the compressed problem by brute force, thus getting a fixed parameter algorithm. However, such a strategy does not work, since in the compression algorithm the witness is allowed to grow by a factor of $poly(n, \log m)$, and therefore solving the compressed problem by brute force may require a super-polynomial time in m. Moreover, even if the witness does not grow, in many cases the witness size depends on the instance size and not on the parameter alone (e.g. in the Clique problem if the parameter is the clique size k then the witness length is $n = k \log m$) thus making the above strategy altogether irrelevant.

Chapter 7 of the monograph of Niedermeier [65] contains several examples of polynomial size kernelizations (e.g. for the languages 3-Hitting Set and Dominating Set on planar graphs). These algorithms yield compression algorithms for the respective languages. We describe one additional example of a compression algorithm that is derived in this manner.

Minimum Fill-In: The minimum fill-in problem is an $\mathcal{NP}$-hard problem that takes as input a graph G and a parameter k, and asks whether there exist at most k edges that can be added to the graph that would turn it into a chordal graph, i.e. one with no induced cycles of length more than 3. This problem has applications in ordering a Gaussian elimination of a matrix.

Claim 2.2 *The minimum fill-in problem with parameter k has witness retrievable compression.*

Proof: Kaplan, Shamir and Tarjan [48] prove that this problem is fixed-parameter tractable (recall that this notion of tractability in parameterized complexity means that the problem is polynomial-time solvable when k is sufficiently small, and in particular for all fixed k). Their algorithm partitions the graph into two sets of nodes A and B where A is of size k^3 and there are no chordless cycles (i.e. an induced cycle of length greater than 3) in G that contain vertices in B. The complexity of this partition is $O(k^2|V||E|)$. They then prove that G has a k edge fill-in if and only if the graph induced by A has a k edge fill-in.

Thus the compression algorithm follows the same partitioning and stores only the graph induced by the small set A. The new graph has at most k^3 vertices and thus can be presented by only $poly(k) \log |k|$ bits. The fill-in for the new instance is exactly that of the original instance and thus the compression can be witness retrievable if the original labels of the vertices are used for the compressed graph as well. $\square$

2.1.1 Sparse languages

We call a language *sparse* if the language contains only of a small fraction of the words under consideration. More precisely:

Definition 2.3 *[Sparse Language] Let L be an $\mathcal{NP}$ language with instance length m and parameter n and denote $L_m = \{x \in \{0,1\}^m \mid x \in L\}$, then L is sparse if there exists a polynomial $p(\cdot)$ such that for all m, n it holds that $|L_m| \le 2^{p(n)}$.*

We show that all sparse languages can be compressed to a size that is dominated by the number of words that are actually in the language. This is shown by a generic compression algorithm for any sparse language. The idea is to apply a random hash function to the instance where the output of the hash is of length $2p(n)$ and thus substantially smaller than m. The new language contains all words that are hashed values of a word in the original language. We note that the compressed language L' lies in $\mathcal{NP}(poly(m))$ (recall that $\mathcal{NP}(poly(m))$ stands for nondeterministic-time $(poly(m))$). In particular, it is not necessarily witness retrievable.

Rather than formally presenting the method for a general sparse language, we describe the method via a sparse language that we call PRG-output. Note that for this language the method is witness retrievable.

Example 2.4 (PRG-Output) *Let G be a pseudorandom generator stretching an n bit seed to an m bit output (with $m >> n$). Define the language* PRG-output *over inputs $y \in \{0,1\}^m$ as*

$$L_G = \{y| \textit{ there exists an } x \textit{ s.t. } G(x) = y\}$$

The language PRG-output is hard to solve for random instances as long as the underlying PRG is secure. Yet it has a simple compression algorithm. Note that simply saving, say, the first $2n$ bits of the instance y is insufficient because if y only differs from $G(x)$ in one bit, then this bit may be anywhere in the m bits.

Claim 2.5 *There exists a witness retrievable compression algorithm for PRG-output.*

Proof: Let $\mathcal{H}$ be a family of almost pairwise independent hash functions from m bits to $2n$ bits. The compression algorithm simply chooses a random $h \in \mathcal{H}$ and outputs $(h(y), h)$. The new language is $L'_G = \{(z,h)| \text{ there exists an } x \text{ s.t. } h(G(x)) = z\}$.

Naturally, if $y \in L_G$ then also $(h(y), h) \in L'_G$ with the same witness (and thus the witness retrievability). On the other hand, if $y \notin L_G$ then by the properties of $\mathcal{H}$, for every seed x we have that $\Pr_h[h(G(x)) = h(y)] < O(2^{-2n})$, and by a union bound over all $x \in \{0,1\}^n$, we get $\Pr_h[h(y) \in L'_G] < O(2^{-n})$. Finally, since there are almost pairwise independent hash functions whose description is of length $O(n) + \log m$ (for example see [59]), then the algorithm is indeed compressing. $\Box$

Sparse subset sum: We show another example of a compressible language called sparse subset sum that is sparse in a different sense than that of Definition 2.3. While the generic compression for sparse languages does not work for this language, it is compressible in its own right. Moreover, the compression algorithm for sparse subset sum is better than the generic algorithm in the sense that the compressed language in the specialized algorithm is in $\mathcal{NP}(poly(n, \log m))$ (or actually in $\mathcal{NP}$) rather than in $\mathcal{NP}(poly(m))$.

Example 2.6 (Sparse Subset Sum) *The language* sparse subset sum *takes as input n values $x_1, \ldots x_n$ each in $\{0,1\}^m$ (with $m >> n$) and a target value $T \in \{0,1\}^m$. An input is in the language if there is a subset $S \subseteq [n]$ where $\sum_{i \in S} x_i = T$ (the sum is taken modulo 2^m).*

Claim 2.7 *There exists a witness retrievable compression algorithm for sparse subset sum.*

Proof: To compress an instance of sparse subset sum simply pick a large random prime $2^n < P < 2^{2n + \log m}$ and store the numbers $y_i = x_i \mod P$ (for every $i \in [n]$), the target $T_P = T \mod P$ and P (the idea of picking a prime P and working modulo P has been useful various applications, e.g. in the Karp-Rabin string matching algorithm [50]). The compressed instance is of length $O(n(n + \log m))$ and the compress language is also subset sum (modulo P). If there exists a set S for which $\sum_{i \in S} x_i = T$ then also $\sum_{i \in S} y_i = T_P$ (hence the witness retrievability). On the other hand, we want that if the original instance was not in the language then for any subset S it will hold that $\sum_{i \in S} y_i \neq T_P$. In order to get $\sum_{i \in S} y_i = T_P$ it is required that P is a divisor of $D = \sum_{i \in S} x_i - T$. However D has at most m/n prime divisors that are greater than 2^n, while the prime P is taken from a range containing $O(2^{2n}m/n)$ primes. Therefore, for every S it holds that $\Pr_P[\sum_{i \in S} y_i = T_P] \leq 2^{-2n}$ and by a union bound over all sets S, the probability of an error is bounded by 2^{-n}. $\Box$

2.2 W-Reductions and Compression-Completeness

The few examples of compression that we have showed clearly indicate that the study of $\mathcal{NP}$ problems with respect to compression gives a distinct perspective, different from the traditional study of $\mathcal{NP}$. The reason is that the typical Karp-reduction between $\mathcal{NP}$ problems does not distinguish between the length of the witness and the length of the instance. For example, when reducing SAT to the Clique problem, one builds a large graph from a CNF formula and asks whether or not it has a Clique of size k. However, in this new instance, the witness size[12] is a polynomial in m (the length of the SAT formula) rather than n (the number of variables in the formula). Thus, it is not clear how to use a compression algorithm for Clique to get a compression algorithm for SAT.

W-reductions and compression-completeness: In order to show that a compression algorithm for L' implies a compression algorithm for L, a more restricted type of reduction is needed. We call this a W-reduction and it is similar to a Karp-reduction but asks an extra property on the length of the witness.

Definition 2.8 (W-Reduction) *For two $\mathcal{NP}$ languages L and L' we say that L* W-reduces *to L' if there exist polynomials p_1 and p_2 and a polynomial time computable function f that takes an instance x for L and outputs an instance $f(x)$ for L' such that:*

1. *$f(x) \in L'$ if an only if $x \in L$.*
2. *If x is of length m with witness length n, then $f(x)$ is of length $p_1(n, m)$ with witness length only $p_2(n, \log m)$.*

We first note that this reduction composes, that is:

Claim 2.9 *If L W-reduces to L' and L' W-reduces to L'' then L W-reduces to L''.*

We next claim that W-reduction indeed fulfills its goal with respect to compression:

Claim 2.10 *Let L and L' be $\mathcal{NP}$ languages such that L' W-reduces to L. Then given a compression algorithm for L, one can obtain a compression algorithm for L'.*

Proof: Suppose that x is an instance for language L' of length m with witness length n. The compression algorithm for L' runs as follows: First use the W-reduction to L and get an instance $f(x)$ for L, and then run the compression algorithm for L on $f(x)$. By the properties of the reduction $f(x)$ is of length $m' = p_1(n, m)$ with witness length $n' = p_2(n, \log m)$. The outcome of the compression is therefore of length $poly(n', \log m') = poly(n, \log m)$. Furthermore, this outcome is in some $\mathcal{NP}$ language L'' if and only if $f(x) \in L$ which in turn happens if and only if $x \in L'$. Thus the combined process gives a compression algorithm for instances of L'. $\Box$

[12]The witness for Clique is a choice of k vertices from the graph

We remark that in the complexity discussion of compression we choose to ignore the issue of witness retrievability. Nevertheless, in order for the W-reduction to relay this property, the reduction itself must also have a witness retrievability property. That is, given a witness w for $x \in L$ then one can efficiently compute w' for $f(x) \in L'$ (without the knowledge of x). We define complete problems with respect to compression: these are defined similarly to the standard notion, but with respect to W-reductions.

Definition 2.11 (Compression-Complete) *A problem L is compression-complete for class $\mathcal{C}$ if:*

1. $L \in \mathcal{C}$
2. *For every $L' \in \mathcal{C}$ the language L' W-reduces to L.*

A language is called compression-hard *for class $\mathcal{C}$ if only requirement 2 holds.*

The relevance of compression-complete problem is stated in the following simple claim.

Claim 2.12 *Let L be compression-complete for class $\mathcal{C}$, then given a compression algorithm for L, one can obtain a compression algorithm for any $L' \in \mathcal{C}$.*

The proof follows directly from the definition of completeness and Claim 2.10.

2.3 The $\mathcal{VC}$ Classification

We now introduce the new classification arising from the study of compressibility of $\mathcal{NP}$ problems. For this we define a series of $\mathcal{NP}$ languages. Some notation: by a **circuit of depth** k we mean a depth k alternating AND-OR circuit where the fan-in of the gates is bounded only by the size of the circuit and negations are only on the input variables (no NOT gates).

Definition 2.13 (Depth$_k$CircuitSAT)
For any $k \geq 2$ consider the $\mathcal{NP}$ problem called Depth$_k$CircuitSAT*:*
Input: *a circuit C of size m and depth at most k over n variables.*
Membership: *$C \in$ Depth$_k$CircuitSAT if there exists a satisfying assignment to C.*

The next language, LocalCircuitSAT, is a less natural one. It is designed to capture computations that do not need to access the whole input, but can rather check only a sub-linear fraction of the input (a good example is verifying that a set of vertices in a graph is indeed a Clique). Let x be a string of length m, if $I = (i_1, \ldots, i_n)$ is a list of n locations in x then we denote by $x(I)$ the values of x at these locations.

Definition 2.14 (LocalCircuitSAT)
Input: *A string x of length m and a circuit C over $n + n \cdot \log m$ variables and of size $(n + n \cdot \log m)$.*[13]
Membership: *If there exists a list I of n locations in x such that $C(x(I), I) = 1$.*

We can now introduce our classification of $\mathcal{NP}$ problems:

Definition 2.15 (The $\mathcal{VC}$ classification of $\mathcal{NP}$ problems) *Consider $\mathcal{NP}$ problems where m denotes the instance size and n denotes the witness size. We define the class $\mathcal{VC}_k$ for every $k \geq 0$. The definition is divided into three cases:*

[13]The choice of the circuit to be of size n' (over n' variables) is arbitrary and other polynomial functions suffice as well. Furthermore, such a circuit of small size may be meaningful since not all the variables have to be used and some might be just dummy variables.

- $k = 0$*: The class* $\mathcal{VC}_0$ *is the class of all languages that admit compression algorithms.*
- $k = 1$*: The class* $\mathcal{VC}_1$ *is the class of all languages that W-reduce to LocalCircuitSAT.*
- $k \geq 2$*: The class* $\mathcal{VC}_k$ *is the class of all languages that W-reduce to Depth*$_k$*CircuitSAT.*

For any function $k(m,n)$ *(where* $k(m,n) \leq m$*) also define* $\mathcal{VC}_{k(m,n)}$ *as the class of all languages that W-reduce to Depth*$_{k(m,n)}$*CircuitSAT. Finally, define* $\mathcal{VC} = \mathcal{VC}_m$ *(the class for* $k(m,n) = m$*).*

A first observation is that simply by definition, the languages LocalCircuitSAT and Depth$_k$CircuitSAT are compression-complete for their respective classes. The most notable example is for the class $\mathcal{VC} = \mathcal{NP}$ where the complete problem is CircuitSAT (satisfiability of a polynomial size circuit).

When discussing a W-reduction to a depth k circuit, we can actually assume without loss of generality that the top gate of this circuit is an AND gate. An immediate corollary is that SAT (that is, satisfiability of CNF formulas) is compression complete for the class $\mathcal{VC}_2$. Formally, let Depth$_k$CircuitSAT$_{AND}$ denote the language Depth$_k$CircuitSAT when restricted to circuits where the top gate is an AND gate.

Claim 2.16 *For any* $k \geq 2$*, we have that a language* $L \in \mathcal{VC}_k$ *if and only if* L *W-reduces to the language Depth*$_k$*CircuitSAT*$_{AND}$.

Proof: We show that any instance that contains a circuit where the top gate is an OR W-reduces to an instance with top gate AND. We prove this first for $k \geq 3$. Denote the input circuit by $C = \bigvee_j \bigwedge_t C_{j,t}$ where each $C_{j,t}$ is a top OR depth $(k-2)$ circuit. If C is satisfiable then $\bigwedge_t C_{j,t}$ is satisfiable for at least one choice of j. Add to the witness the index i of this satisfiable sub-circuit (i is given by the boolean variables $i_1, ..., i_\ell$ where ℓ is logarithmic in $poly(m,n)$). For each j, denote $C'_{j,t} = C_{j,t} \vee i_1^{\bar{j}_1} \vee ... \vee i_\ell^{\bar{j}_\ell}$, where $i^{\bar{j}}$ denotes $i \oplus j$. Notice that $C'_{j,t}$ is always satisfied for $j \neq i$, and for $j = i$ is satisfied if and only if $C_{i,t}$ is satisfied. Thus the circuit can now be written as $C' = \bigwedge_{j,t} C'_{j,t}$ that is satisfiable if and only if the original circuit was. The top OR gate of C is therefore removed in the new instance C' while adding only a small number of variables, thus the input to the circuit witness remains of order $poly(n, \log m)$ as required.

In the case $k \geq 3$, the depth of the new instance becomes $k-1$. In the case that $k = 2$, the bottom level that included only variables is transformed into an OR of variables, thus the new circuit is simply a CNF formula (and the depth remains $k = 2$). $\square$

The $\mathcal{VC}$ Hierarchy: The $\mathcal{VC}$ classification indeed defines a hierarchical structure. That is:

$$\mathcal{VC}_0 \subseteq \mathcal{VC}_1 \subseteq \mathcal{VC}_2 \subseteq \mathcal{VC}_3 \cdots \subseteq \mathcal{VC}.$$

And in general, for every two polynomially bounded functions $k(n,m), \ell(n,m)$ such that for all n,m we have $k(n,m) \leq \ell(n,m)$ then $\mathcal{VC}_k(m,n) \subseteq \mathcal{VC}_\ell(m,n)$. Furthermore, $\mathcal{VC} = \mathcal{NP}$ by the definition of $\mathcal{NP}$. These observations follow trivially by the definitions, the only non-trivial part being the fact that $\mathcal{VC}_1 \subseteq \mathcal{VC}_2$, that is proved next.

Lemma 2.17 $\mathcal{VC}_1 \subseteq \mathcal{VC}_2$

Proof: We need to show a W-reduction from LocalCircuitSAT to SAT. The input is therefore a long string x and small circuit C on $n + n\log m$ variables. Let $i_1, ...i_n$ denote the potential locations in the string that the circuit C receives as inputs. Also define the variables $y_1, ..., y_n$ to indicate the values of x in the corresponding locations (that is $y_t = x_{i_t}$ for $t \in [n]$). Thus the circuit C runs on the variables $y_1, ..., y_n$ and the bits of $i_1, ...i_n$.

We first note that C is of size $p(n, \log m) = (n + n \log m)$ and may be reduced (via Cook's Theorem [17]) to a CNF formula Φ_C over $O(p(n, \log m))$ variables and of size $O(p(n, \log m))$ that is satisfiable if and only if C is satisfiable.

Thus we have a CNF formula over the variables $y_1, ..., y_n, i_1, ...i_n$ and some extra variables. This formula's satisfiability will be equivalent to the membership of the LocalCircuitSAT instance if we manage to force the variables of y to take the values $y_t = x_{i_t}$. This is done by adding additional clauses to the CNF in the following manner: For simplicity we describe this only for y_1, where the same is repeated for every other y_t for $t \in [n]$. Define for each $j \in [m]$ a formula $\Phi_j = (y_1 = x_j) \vee (i_1 \neq j)$. Notice that $\Phi_{i_1} = 1$ if and only if $y_1 = x_{i_1}$. Denote the bits of i_1 by $i_{1,1}, ..., i_{1,d}$ where $d = \lceil \log m \rceil$. An alternative way to write Φ_j is as the following CNF (recall that $i^{\bar{j}}$ denotes $i \oplus j$):

$$\Phi_j = (y_i \vee \overline{x_j} \vee i_{1,1}^{\bar{j_1}} \vee ... \vee i_{1,d}^{\bar{j_d}}) \wedge (\overline{y_i} \vee x_j \vee i_{1,1}^{\bar{j_1}} \vee ... \vee i_{1,d}^{\bar{j_d}})$$

Finally, to force $y_1 = x_{i_1}$ we simply take the new CNF to be $\Phi_C \wedge \bigwedge_{j \in [m]} \Phi_j$. The same is repeated to force $y_t = x_{i_t}$ for all $t \in [n]$. $\square$

2.4 The $\mathcal{VC}$ Classification and Verification with Preprocessing

We now discuss the $\mathcal{VC}$ hierarchy from a different angle, that of the verification complexity of a language. This approach, though slightly more cumbersome than the definition via W-reductions, gives more intuition as to what it means to be in a class $\mathcal{VC}_k$. The new view defines the $\mathcal{VC}$ hierarchy with respect to the verification algorithm for L, that is, the efficient procedure that takes a witness w for $x \in L$ and verifies that it is indeed a true witness. We point out that the nature of verification algorithms may vary when discussing different $\mathcal{NP}$ problems. For example, in the k-Clique problem the verification algorithm needs to check only $O(k^2)$ edges in the graph, and thus can read only a sub-linear part of the instance. In SAT, on the other hand, all the clauses in the formula must be checked when verifying a witness.

Simply looking at the verification algorithm of a language is not sufficient. For starters, classification according to verification does not distinguish between problems in $\mathcal{P}$ that are trivially compressible and $\mathcal{NP}$-complete languages. Instead, we consider the notion of verification with preprocessing. This is the process for verifying that $x \in L$ when given a witness, that also allows a preprocessing stage to the instance. Formally:

Definition 2.18 (Verification with Preprocessing) *Let L be an $\mathcal{NP}$ language with instances of length m and witness length n. A pair of* polynomial time *algorithms (P, V) are called a* verification with preprocessing *for L if the following two step verification holds:*

1. **Preprocessing:** *P gets an instance x and outputs a new instance $P(x)$.*
2. **Verification:** *There exists a polynomial $p(\cdot, \cdot)$ such that $x \in L$ if and only if there exists a witness w of length at most $p(n, \log m)$ such that $V(P(x), w) = 1$.*

Notice that when allowing for preprocessing, then all problems in $\mathcal{P}$ have a pair (P, V) where P solves the problem and stores the answer while V simply relays this answer. Thus when considering the complexity of V in this definition, then easy problems indeed have very low complexity.

The $\mathcal{VC}$ Classification via Verification with Preprocessing: An alternative and equivalent way to view the classes in the $\mathcal{VC}$ hierarchy is based on the verification algorithm V in a verification with preprocessing pair (P, V). The problems are divided into two families:

- The class $\mathcal{VC}_1$ is the set of the languages that have very efficient verification (i.e. $poly(n, \log m)$ rather than $poly(n, m)$). We assume random access to the instance, thus such a verification algorithm only accesses a sub-linear fraction of the instance.

- The languages whose verification is not very efficient (run in time $poly(n, m)$). This family is further classified into sub categories. The class $\mathcal{VC}_k$ is the class of languages where the verification algorithm V has a representation as a depth k polynomial size circuit (polynomial in n and m).

This definition is equivalent to the definition via W-reductions since the W-reduction to the complete problem can simply be viewed as the a preprocessing stage. In the other direction, every preprocessing stage is actually a W-reduction to the language defined by V.

It is interesting to note that Buss and Islam [10] give an alternative view with similar flavor to the *Weft* hierarchy of parameterized complexity. They call it "prepare, guess and check" in which they essentially add a preprocessing phase to a previous approach of Cai and Chen [12].

2.5 Within $\mathcal{VC}_1$ - The class $\mathcal{VC}_{OR}$

Arguably, the most interesting class in the hierarchy is the bottom class $\mathcal{VC}_1$. It contains many natural problems such as Clique or small subset-sum[14] that only test local properties of the input. Furthermore, it is presumably the easiest to find compression algorithms for. We further refine our hierarchy within the class $\mathcal{VC}_1$ by introducing another class, the class $\mathcal{VC}_{OR}$. Consider the language OR(L) that take a large OR of small instances of a language L. Formally:

Definition 2.19 (OR(L))
Let L be an $\mathcal{NP}$ language. Define the language OR(L) as follows

Input: *m instances $x_1, ..., x_m$ to the language L, each of size n.*

Membership: *If there exists $i \in [m]$ such that $x_i \in L$.*

Specifically the language **OR(CircuitSAT)** *is defined as:*

Input: *m different circuits where each circuit is of size n.*

Membership: *If one of the m circuits is satisfiable.*

This language is used to define the following class:

Definition 2.20 *The class $\mathcal{VC}_{OR}$ is the class of all languages that W-reduce to OR(CircuitSAT).*

We first note that in each of the m small instances, the instance length and witness length are polynomially related. So unlike the general case where we focused only on short witness languages, when talking about OR(L), any language $L \in \mathcal{NP} \backslash \mathcal{P}$ is interesting. For example, the language OR(3-SAT) is not trivially compressible. Moreover, it is compression-complete for $\mathcal{VC}_{OR}$.

Claim 2.21 *Let L be any $\mathcal{NP}$-complete language, then OR(L) is compression-complete for $\mathcal{VC}_{OR}$.*

Proof: The W-reduction from OR(CircuitSAT) to OR(L) simply runs the standard Karp reduction from CircuitSAT to L for each of the m circuits independently. The witness for each circuit was of at most n and is now of size $p(n)$ for some polynomial p. In addition the witness contains an index of the instance of L that is satisfied, thus the total witness length is $p(n) + \log m$. $\square$

[14]This problem takes m values and a target value and asks if there is a small (size n) subset of the values that adds up to the target.

For example, the problem $OR(Clique)$ that gets m small graphs (over n vertices) and asks whether at least one of the graphs has k sized clique (where $k = O(n)$) is also compression-complete for $\mathcal{VC}_{OR}$. Moreover, we note the following claim that is relevant to our cryptographic applications (in Sections 5 and 4):

Claim 2.22 *Clique is compression-hard for $\mathcal{VC}_{OR}$.*

Proof: The language $OR(Clique)$ W-reduces to Clique simply by taking one graph that is the union of all the small graphs in the $OR(Clique)$ instance. Clearly there is a clique in the union if and only if there is a clique in at least one sub-graph. $\square$

A similar claim is true for all problems involving searching for a connected subgraph of size n in a graph of size m as long as the problem of deciding whether a graph of size $p(n)$ contains such a subgraph is NP-Hard for some polynomial $p(\cdot)$. This is true, for instance, for the problem of whether there is a path of length n.[15] On the other hand we have that:

Claim 2.23 $\mathcal{VC}_{OR} \subseteq \mathcal{VC}_1$

Proof: This is best seen by W-reducing OR(Clique) to LocalCircuitSAT. Given graphs $G_1, ..., G_m$ for OR(Clique), generate the instance $x = G_1, ..., G_m$ and a circuit C that receives the locations of a clique in one of the graphs and checks whether they are indeed the edges in these locations form a clique (all belong to the same graph and are the edges induced by k vertices etc...). The size of the circuit is $p(n, \log m)$ for some polynomial p since it checks only locations in x that belong to one graph (of size n). Finally, add $p(n, \log m)$ dummy variables to the circuit so that the circuit C has size becomes equal to the number of input variables (as is required in LocalCircuitSAT). $\square$

Furthermore, $\mathcal{VC}_0 \subseteq \mathcal{VC}_{OR}$, since any compressible language can be W-reduced by the compression algorithm to a language with instance size $p(n, \log m)$ and this instance can reduced to CircuitSAT and viewed as an OR of a single small circuit and hence is in $\mathcal{VC}_{OR}$. Note that here too, one may need to add dummy variables to keep the Circuit quadratic in its input. Altogether we have that:

$$\mathcal{VC}_0 \subseteq \mathcal{VC}_{OR} \subseteq \mathcal{VC}_1.$$

2.6 The $\mathcal{VC}$ Classification and some $\mathcal{NP}$ Problems

In general, most of the $\mathcal{VC}$ classification focuses on W-reductions to depth k circuits. The reasoning for this is that there is a certain tradeoff between depth and the number of variables. More precisely, we can reduce the depth of a verification circuit, but only at the price of adding additional variables (this is done using methods from Cook's Theorem [17] and requires adding a variable for each gate in one intermediate level of the circuit). Since the number of variables is the focal point when discussing compression (as it coincides with the witness size), then depth turns out to be central in our classification.

Given our current state of knowledge, there are a few plausible views of the world. The two *endpoint* scenarios are (i) there is compression for every language in $\mathcal{NP}$ (as would be implied by a compression algorithm for CircuitSAT), (ii) there is only compression for a few select problems, such as the examples in section 2.1. A third option is that there is a compression algorithm for some compression-complete problem in the hierarchy (say for $\mathcal{VC}_k$), which would imply the collapse of all the classes below $\mathcal{VC}_k$ to $\mathcal{VC}_0$.

We will briefly go over a few key classes in the hierarchy and a few examples of natural $\mathcal{NP}$ problems and their classification (as we know it) within the $\mathcal{VC}$ hierarchy:

[15] It is interesting to note that whereas the problem of finding a path of length n is fixed parameter tractable [3], Feige and Kilian [32] give indications that the Clique problem is hard for small n (via subexponential simulations).

The class $\mathcal{VC}_0$: Currently we know that this class contains all the languages in $\mathcal{P}$, languages that are already compressed by definition (such as 3-SAT), and the languages that we showed compression algorithms to (Vertex-cover, PRG-output and Minimum-fill-in).

The class $\mathcal{VC}_{OR}$: This class contains all languages OR(L) for an $\mathcal{NP}$ language L. One natural example is the OR(SAT) problem which is actually a depth 3 circuit where the fan-in at the two bottom levels is bounded by n and only the top gate is allowed to be of greater fan-in. Some important languages in this class are those that need to be compressed in the cryptographic applications in Sections 5 and 4.

The class $\mathcal{VC}_1$: Since we are only interested in problems where the witness size n is much smaller than the instance size m, then many natural problems with this restriction are in $\mathcal{VC}_1$. For example, graph problems that ask whether a small graph can be embedded in a large graph are all in $\mathcal{VC}_1$. The Clique problem (with a clique of size n), or Long-Path (a path of length n that does not hit any vertex twice) are such small graph embedding problems. Small Subset-Sum is another natural language in $\mathcal{VC}_1$. This language receives a set of m values and a target sum and asks whether there is a small (size n) subset for which the values add up exactly to the target sum.

Dominating Set: The problem asks, given a graph, whether there is a set of k vertices such that all the graph is in its neighbor set. Dominating set is in the class $\mathcal{VC}_3$ as implied by the following verification: the witness is a set S and the algorithm tests that $\forall$ vertex v $\exists$ vertex $u \in S$ such that (u, v) is in the graph. The $\forall$ translates to and AND gate and the $\exists$ translates to an OR gate. Finally, testing that an edge is in the graph requires an AND over the $O(\log m)$ bits representing this edge. In total, this is a depth 3 circuit. Note that a straightforward verification of vertex cover will also yield a depth 3 circuit. However, while vertex cover is compressible and in $\mathcal{VC}_0$, for dominating set we are unaware a better method. In addition, dominating set is *compression-hard for* $\mathcal{VC}_2$. This is seen by a standard reduction of SAT to dominating set in which a SAT formula with n variables and m clauses is transformed into a graph with $m + 3n$ vertices with the property that the graph has a dominating set of size n iff the SAT formula is satisfiable.[16]

Weighted-SAT: Given a CNF formula of length m the problem asks if it has a satisfying assignment of weight at most k (at most k variables are assigned the value 1). Unlike our previous discussions of SAT, the number of variables here is only bounded by m and the short witness simply consists of the list of all variables that receive the value 1 (that is, the witness is of length $n = k \log m$). This problem, with constant clause size, serves as the basic complete problem for the parameterized complexity class $W[1]$, which is at the bottom of the W-hierarchy (see [24]). However, with regards to compressibility, we only know how to place it in the class $\mathcal{VC}_4$. This is shown by the following verification procedure (using the same logic as with Dominating-Set): For every witness (list) L, the algorithm tests that $\forall$ clauses C either $\exists$ a variable $x \in C$ such that $x \in L$ or $\exists$ a negated variable $\bar{x} \in C$ such that $x \notin L$. The verification of $x \in L$ adds up to total depth 3 by testing that $\exists y \in L$ such that $x = y$ (where $x = y$ is tested by an AND over the bits of x and y). On the other hand, verifying that $x \notin L$ requires total depth 4 as it runs $\forall y \in L$ we have $x \neq y$. The overall depth is thus dominated by the negated variables and is thus 4.

OR of (large) instances: Consider the Or of CNF formulas over few variables (each CNF formula may be large, unlike in the language $OR(SAT)$ where the CNF formulas are considerably smaller than the

[16] In a nutshell, the reduction creates a triangle for each variable x_i of the formula. One of the nodes of the triangle is identified with the positive variable and another with its negation while the third is connected only to the other two. In addition, a vertex is created for each clause in the formula. Now, each literal is connected with all of the clauses that it appears in. The generated graph has a dominating set of size n iff the formula is satisfiable.

fan-in of the OR gate). In other words, instances of this language are depth three circuits where the top gate is an Or gate. Yet the language is actually in $\mathcal{VC}_2$, as implied by Claim 2.16.

Integer Programming (IP): An instance of integer programming consists of a list of m linear constraints on n integer variables with the goal of maximizing a linear target function over these n variables (under the list of constraints). Unlike its counterpart of linear programming, where the variables may take real values and is polynomial time solvable, integer programming is $\mathcal{NP}$-hard even when the variables are restricted to taking only the values 0 and 1 (one of Karp's original problems [49]). Thus, the decision variant of integer programming, where the number of constraints is much larger than the number of variables, is interesting with respect to compression. First, compressing it is at least as hard as compressing SAT: given a SAT instance with n variables and m constraints it is simple to come up with a corresponding IP instance with $2n$ variables and m constraints, i.e. IP is $\mathcal{VC}_2$-hard. On the other hand, a straightforward verification of a witness for this problem takes the proposed assignment for the n variables and checks if it satisfies each of the constraints. The verification of a linear constraint can be achieved in logarithmic depth (in n), placing IP in $\mathcal{VC}_k(n)$ for $k(n) = \Omega(\log n)$. We are unaware of a (significantly) better classification (of lower depth) for integer programming.

2.7 On Compression of Search Problems

So far, the $\mathcal{NP}$ problems that we discussed were all decision problems, that is, they ask if $x \in L$, and are answered by YES or NO. When considering a specific $\mathcal{NP}$ relation R_L associated[17] with L, then the above decision problem has a natural search problem associated with it, which is to actually find a witness to $x \in L$ with respect to the relation R_L. A solution to such a problem is an n bit string rather than just a single bit.

Loosely speaking, a compression algorithm for the search instance should produce a shorter output that contains enough information about some witness for the original problem.

Definition 2.24 (Compression for search problem) *A compression algorithm for an $\mathcal{NP}$ search problem L (with respect to R_L) is a pair of algorithms (Z, E) with a polynomial $p(\cdot,\cdot)$, where Z is a polynomial time compression algorithm and E is an unbounded extraction algorithm. Given an instance x with witness parameter n we should have that:*

1. *$Z(x)$ is of length at most $p(n, \log m)$.*
2. *If $x \in L$ and there is a witness of length n, then $E(Z(x)) = w$ where w is a witness to $x \in L$ with respect to R_L.*

It is natural to consider the relationship between the difficulty of decision and search for a given problem, as was done in other settings such as average-case complexity by Ben-David et al. [8]. We show that for any problem a compression for the decision variant also yields a compression for the search variant, *without an increase in the VC hierarchy*.

Theorem 2.25 *For any $k \leq 1$, if the class $\mathcal{VC}_k$ has a compression algorithm, then there is a compression algorithm for the search problem of a relation R_L of $L \in \mathcal{VC}_k$. This is true also for $\mathcal{VC}_{OR}$.*

The technique of the proof below also comes in handy in proving Theorem 4.4, regarding the application of the ability to compress, say SAT, to cryptanalysis in hybrid bounded storage model. In the following proof, a witness to $x \in L$ refers to a witness according to the specific relation R_L associated with L.

[17] Let L be an $\mathcal{NP}$ language with parameters m and n. A relation R_L associated with L is a polynomial time function $R_L : \{0,1\}^m \times \{0,1\}^n \to \{0,1\}$, such that for every $x \in \{0,1\}^m$ it holds that $x \in L$ iff there exists a $w \in \{0,1\}^n$ such that $R_L(x,w) = 1$.

Proof: Given an instance x to a language L, for any $i \in [n]$, consider the $\mathcal{NP}$ problem L_i that asks whether there exist an n bit witness w to $x \in L$ such that $w_i = 1$ (the i^{th} bit of w is 1). The language L_i is *also* in $\mathcal{VC}_k$ since its verification circuit is the same as the one for L with an additional AND to the variable w_i (this AND gate is incorporated into the top level AND of the circuit thus the depth remains k).

Our first attempt is to compress the instance x for every $i \in [n]$ with respect to the language L_i (denote such a compression by $Z_{L_i}(x)$). Thus we store $Z_{L_i}(x)$ for all $i \in [n]$, which amounts to $n \cdot p(n, \log m)$ bits, for some polynomial $p(n, \log m)$ (this is also in $poly(n, \log m)$). Now suppose that there is only a *single* witness w to x; then one can extract w bit by bit, by solving the compressed instance of each bit. However, this fails when w is not the only witness, and we might obtain inconsistent answers for the different bits.

The natural idea now is to use the reduction of Valiant and Vazirani [74] to a unique witness, as was done by Ben-David et al. [8] for showing that average NP being in BPP implies also a randomized search algorithm for average NP. The idea is to choose a pairwise-independent hash function h that is appropriately shrinking, and add to the language the requirement that $h(w) = 0$. We use the following lemma:

Lemma 2.26 ([74]) *Let L be an $\mathcal{NP}$ language and for every x denote by W_x the set of all witnesses to $x \in L$. Let ℓ be such that $2^\ell \leq |W| \leq 2^{\ell+1}$. Let $\mathcal{H}_{\ell+2}$ be a family of pairwise independent hash functions with $h : \{0,1\}^n \to \{0,1\}^{\ell+2}$ for all $h \in \mathcal{H}_{\ell+2}$. Then*

$$\Pr_{h \in \mathcal{H}_{\ell+2}}[|\{w : w \in W_x \textit{ and } h(w) = 0\}| = 1] \geq \frac{1}{8}$$

Let $\mathcal{H}$ be a family of pairwise independent hash functions. Consider the $\mathcal{NP}$ language $L^{\mathcal{H}}$ whose elements are of the form (x, h) where $h \in \mathcal{H}$ maps strings of length n to some shorter length. We have that $(x, h) \in L^{\mathcal{H}}$ if there is a witness w for $x \in L$ and $h(w) = 0$. We note that this language is also in $\mathcal{VC}_k$, since the additional requirement that $h(w) = 0$ can be verified efficiently over n variables (the hash function h computation is efficient). By Cook's theorem this computation may be represented as a CNF formula ϕ_h over these variables plus only $poly(n)$ additional variables. Thus adding the requirement of the hash does not add to the depth of the verification circuit for L. This is easy to for $\mathcal{VC}_k$, and for $\mathcal{VC}_{OR}$ note that we can add (conjunction) the CNF formula ϕ_h to each instance of CircuitSAT, while keeping the problem in $\mathcal{VC}_{OR}$.

Now, if we enumerate on all values of ℓ, then with probability at least $\frac{1}{8}$, for the correct ℓ we will get that $L^{\mathcal{H}}$ has a unique witness; storing $Z_{L_i^{\mathcal{H}}}(x, h)$ for all i suffices to maintain the information about this witness. This can be repeated sufficiently many times (say $O(n)$ times), so that with overwhelming probability one of the attempts will indeed give a unique witness. However, this solution is also insufficient, since we have stored a list of $O(n^2)$ compressed values ($O(n)$ repetitions for each value of $\ell \in [n]$) with the guarantee that with overwhelming probability one of them is a witness for x, but we do not known which one (recall that we cannot store the original instance and thus cannot verify that a witness is correct).

Our final attempt succeeds in reducing the list of potential witnesses into a unique and true witness. This compression is as follows: Denote by $L_{\bar{i}}$ the language that asks whether there exist an n bit witness w to $x \in L$ such that $w_i = 0$ (similar to L_i but with w_i negated). The compression of an instance x to the search problem L goes as follows:
For every $\ell \in [n]$ repeat the following n times:

- Choose $h \in_R \mathcal{H}_{\ell+2}$.
- For all $i \in [n]$ store $Z_{L_i^{\mathcal{H}}}(x, h)$ and $Z_{L_{\bar{i}}^{\mathcal{H}}}(x, h)$.

The extraction procedure is as follows: For all ℓ and $h \in \mathcal{H}_{\ell+2}$, solve all the compressed instance pairs. For every pair $Z_{L_i^{\mathcal{H}}}(x, h)$ and $Z_{L_{\bar{i}}^{\mathcal{H}}}(x, h)$, if both are negative or both are positive, then ignore all values that are compressed with this h. Only if for all i we have that exactly one of the instances being correct, then output the i^{th} bit of w according to the result.

The above algorithm indeed compresses, since it only adds a factor of n^3 to the overall storage. With probability at least $1 - 2^{-O(n)}$ at least one of the chosen h's is successful in leaving exactly one witness to $x \in L_h$, and this witness will be extracted. Finally, if h did not leave exactly one witness, then this will be identified: If there are no witnesses then $Z_{L_i^{\mathcal{H}}}(x, h)$ and $Z_{L_{\bar{i}}^{\mathcal{H}}}(x, h)$ will both be negative for all i. If there is more than one witness, then for at least one choice of i both $Z_{L_i^{\mathcal{H}}}(x, h)$ and $Z_{L_{\bar{i}}^{\mathcal{H}}(x,h)}$ will be positive. $\square$

2.8 On Maintaining Other Information

We have seen that compression may maintain much more than just a yes/no answer. A natural question to ask what other types of information may be maintained through compression algorithms. The following are some examples:

Number of witnesses: The compression described above actually maintains an approximation of the number of witnesses to $x \in L$ (with respect to R_L). Once the chosen k is too large, there will be a sharp drop in the probability of having a witness and this can be observed when extracting the witnesses and indicate what is the right k.

An almost random witness: The compression above also outputs a witness that is almost uniformly distributed over W_x. Or more accurately, the probability of getting each witness is bounded by a constant times $1/|W_x|$.

On maintaining all witnesses: As opposed to maintaining a single witness or the number of witnesses, a compressed instance cannot always maintain the information about *all* of the witnesses of an input instance. This is shown by the following simple information theoretic argument: encode an m bit string s with a DNF circuit C by constructing for each position $j \in [m]$ a formula C_j on $\log m$ variables. If $s[j] = 1$ then take C_j to be circuit that is satisfied iff the variables encode the index j. If $s[j] = 0$ then C_j is the non-satisfiable circuit $C_j = 0$. The circuit C is formed by taking an OR of all these circuits ($C = \bigvee_{j \in [m]} C_j$). The satisfying assignments of C correspond exactly to the 1's in s. Consider C as an input to the language as CircuitSAT[18]. Suppose that there exists a compression algorithm that maintains all of the witnesses of a circuit C. In particular, this means that the m bit string s may also be extracted from the compressed instance. But this is clearly impossible information theoretically, since m random bits may not be represented by $poly(n, \log m) < m$ bits. So we conclude that if our goal is come up with a compression algorithm for SAT then we must come up with a way of losing information about the witnesses.

In the examples of compression that we have seen in Section 2.1, the compression algorithms for vertex cover, PRG-output and Minimum fill-in actually maintain all the witnesses. On the other hand, the compression for GapSAT (which we will see in Section 2.9) does not necessarily maintain this information, as it is based on sampling.

2.9 Speculation on Compression

We give two arguments that may be viewed as evidence to the existence and non-existence of compression respectively.

[18]The circuit C is actually an instance for the language $OR(CircuitSAT)$.

An Optimistic View - Compression of a promise problem and the PCP Theorem: Consider the promise problem GapSAT that takes as input a CNF formula Φ of size m over n variables and the guarantee that either Φ is satisfiable or it is at most $(1 - \frac{1}{2n})$-satisfiable (no assignment satisfies more than $(1 - \frac{1}{2n})$ of its clauses). The task is to decide if Φ is satisfiable or far from satisfiable.

Such a problem has a simple and witness retrievable compression. The idea is to choose $O(n^2)$ random clauses from Φ and take the AND of these clauses to be the compressed formula Ψ. This compression works because if Φ is far from satisfiable then for every assignment the formula Ψ is satisfied with probability at most 2^{-2n} (Ψ does not contain one of the $\frac{1}{2n}m$ unsatisfied clauses). Taking a union bound over all assignments, we get that with probability $(1 - 2^{-n})$ the formula Ψ has no satisfying assignment. On the other hand, if Φ is satisfiable then the same assignment also satisfies Ψ (and hence the witness retrievability). Note that our definition of GapSAT is robust in the sense that GapSAT is compressible whenever the gap is $(1 - \frac{1}{p(n)})$ *for every choice of a polynomial* $p(\cdot)$.

The above simple compression algorithm is especially interesting in light of the PCP Theorem. One way to view the PCP Theorem is as an efficient reduction from an instance of SAT to an instance of GapSAT. Thus one can hope to combine the PCP reduction with the above compression and get a compression for general SAT. However, reducing general SAT to GapSAT via the PCP is not a W-reduction as the witness size grows to the order of the instance size. For starters, the PCP Theorem is typically defined over 3-CNF formulas, and the reduction of a general size m CNF to a 3-CNF adds $O(m)$ variables. In order for this approach to achieve compression for SAT, we require a new PCP Theorem that is actually a W-reduction.

GapSAT is just one example of a gap problem that admits compression. For instance, one can consider the promise problem GapClique where a graph of size m either has a Clique of size m/n or contains no Clique of size n. As in the case of GapSAT, GapClique is compressible by sampling a subset of its vertices. Thus, coming up with a W-reduction from a general (n', m')-Clique problem (the graph of size m' either contains a clique of size n' or not) to (n, m)-GapClique would enable the compression of Clique. We view finding PCPs that are also W-reductions as a major research direction, especially in light of the recent new proof to the PCP Theorem of Dinur [23].

A Pessimistic View - On Oblivious Compression: We have seen in Section 2.8 that it is impossible to maintain all of the information in an instance when compressing it and some information is necessarily lost (for example the list of all witnesses cannot be kept). On the other hand, we show that if compression exists then it is not likely to lose too much information about the original instance. Such a result would entail the collapse of the polynomial hierarchy to its second level. More formally:

Let Z be a compression algorithm for SAT. We consider it as a two input algorithm taking a formula Φ and local randomness $r \in \{0,1\}^\ell$. Denote by $Z(\Phi, U_\ell)$ the random variable taking the output of Z with fixed input Φ and random $r \in_R \{0,1\}^\ell$. Let $\mathbf{X}$ be a distribution over formulas. The random variable $Z(\mathbf{X}, U_\ell)$ denotes the output of Z under a choice of random r and a random Φ from the distribution $\mathbf{X}$.

The compression Z is said to be *oblivious* if there exists a samplable distribution $\mathbf{X}$ over satisfiable formulas, such that for every satisfiable instance Φ the distribution $Z(\Phi, U_\ell)$ and the distribution $Z(\mathbf{X}, U_\ell)$ are statistically close (within statistical distance ε).

Claim 2.27 *If there exists an oblivious compression for SAT, then the polynomial hierarchy collapses to its second level.*

One way to prove this result is by noticing that oblivious compression for SAT can be used for *instance hiding computation* of SAT in the sense of [2], which in turn entails the collapse of the polynomial hierarchy. Instead we give a direct proof:

Proof: We show that if oblivious compression of SAT instances exists then Co-SAT $\in \mathcal{AM}$. Consider the following interactive proof that an instance $\Phi \notin SAT$. The verifier chooses a random satisfiable formula $\Psi \in \mathbf{X}$ randomness $r \in U_\ell$ and flips a random coin c. If $c = 0$ then the verifier sends $\xi = Z(\Phi, r)$ to the prover, if $c = 1$ he sends $\xi = Z(\Psi, r)$. The prover then answers 1 if the compressed instance is satisfiable and 0 otherwise. The verifier accepts if the provers answer equals his bit c and rejects otherwise.
Completeness: If indeed $\Phi \notin SAT$, then the prover will be able to tell whether the verifier used a coin $c = 0$ or $c = 1$, simply by testing the satisfiability of ξ and replying correctly.
Soundness: Suppose that $\Phi \in SAT$, then by the obliviousness property of Z the message ξ is from nearly the same distribution whether $c = 0$ or $c = 1$ and the prover is bound to error with probability $\frac{1}{2} + \varepsilon$. $\square$

Thus, oblivious compression for SAT is not likely to exist. However, the languages we would like to compress for the applications in Sections 3, 5 and 4 are actually in $\mathcal{NP} \cap Co\text{–}\mathcal{NP}$, and thus for these applications even oblivious compression is actually a valid possibility.

Part II: Cryptographic Applications

3 Basing Collision Resistant Hash Functions on Any One-Way Function

Loosely speaking, a family of length reducing functions $\mathcal{H}$ is called collision resistant hash functions (CRH) if no efficient algorithm can find collisions induced by a random member of the family. That is, no PPTM can find for a randomly chosen $h \in_R \mathcal{H}$, a pair of input strings x and x' such that $x \neq x'$ but $h(x) = h(x')$. In addition we want (i) An efficient algorithm for *sampling* from $\mathcal{H}$ using (possibly secret) randomness (the secret coins approach is potentially more powerful then when only public coins are used [43]) and (ii) An efficient evaluation algorithm that given the description of $h \in \mathcal{H}$ and x produces $h(x)$. As mentioned in the introduction, CRHs have wide cryptographic applications, see discussion and formal definitions in, for example, [47]. We are interested in basing CRH on as general assumption as possible. There is no known construction of CRH from general one-way functions or one-way permutations. Moreover, Simon [71] showed that basing CRH on one-way permutations cannot be achieved using black-box reductions[19]. We show that compression can be used to bridge this gap.

Theorem 3.1 *If there exists an errorless compression algorithm for SAT, or for any problem that is compression-hard for $\mathcal{VC}_{OR}$, then there exists a construction of a family of Collision Resistant Hash functions (CRH) based on any one-way function.*

Proof: Let $(\mathrm{C}ommit, \mathrm{V}erify)$ be a statistically binding computationally hiding commitment scheme based on the one-way function f. Recall that the protocol $\mathrm{C}ommit$ takes from the sender a string S and randomness r and after an interaction the receiver gets a commitment σ. The polynomial time algorithm $\mathrm{V}erify$ takes the commitment σ and a possible opening to value S' with randomness r' and verifies that S', r' are consistent with σ. One could take for example the commitment scheme of Naor [60] based on the one-way function f.[20] In our setting we can work under the assumption that the sender (in the commitment) is honest, and in such a case, the commitment may be achieved without interaction at all[21].

[19] Simon's black-box impossibility result [71] is actually stated for the *public* coins version of CRH rather than the *secret* coins variant that we discuss. However this separation also holds for the case of secret coins (as pointed out in [43]).

[20] To be more exact, the commitment of [60] can be based on the pseudorandom generator of Håstad et al. [42] which in turn can be based on the function f.

[21] In the scheme of Naor [60], the receiver is required to provide the sender with a (public) random string. Certainly, an honest sender can generate this string by himself without harming the properties of the commitment. Thus in such a setting, the sender can generate the commitment without interaction.

The CRH construction is inspired by the approach of Ishai, Kushilevitz and Ostrovsky [47] for constructing collision resistant hash from Private Information Retrieval (PIR). A high level description is: choose a hash function from a naive hash family with no computational hardness guarantees; in the construction below we use the selection function, i.e. a random position i. The new hash function is defined by a computationally hiding commitment to the naive hash function, and the output of the new hash function is a compression maintaining the information of the committed naive hash function when applied to the input (i.e. compression of the formula that checks that the value is what it claimed to be). Intuitively, finding a collision would require guessing with non-negligible advantage the naive hash function (the position i). The actual construction is given in Figure 1.

CRH family $\mathcal{H}_f$:

Description of the hash function: Let Z be a compression algorithm for SAT. A function in the CRH collection is denoted h_{σ,r_Z} and defined by a commitment σ to a value $i \in [m]$, and randomness r_Z for Z. The commitment uses security parameter n (where $n << m$).

Input to h_{σ,r_Z}: a string $x \in \{0,1\}^m$

The CNF formula $\Phi_{\sigma,x}$ is defined as follows:

- Denote by $\mathrm{V}erify_\sigma$ the algorithm $\mathrm{V}erify$ with the input σ fixed. That is, $\mathrm{V}erify_\sigma$ takes as inputs y and r and accepts if and only if they form a legal opening of the commitment σ (and in particular this means that $y = i$).
- Translate $\mathrm{V}erify_\sigma$ into a CNF formula Φ_σ over the variables $y_1, ..., y_\ell$ of y and the bits of r (using Cook's reduction).
- For every $j \in [m]$ define the clause $C_{j,x} = (y_1^{\bar{j}_1} \vee y_2^{\bar{j}_2} \vee .... \vee y_\ell^{\bar{j}_\ell})$ if $x_j = 0$ (where y^0 denotes $\bar{x}$ and y^1 denotes x) and $C_{j,x} = 1$ if $x_j = 1$.
- Set
$$\Phi_{\sigma,x} = \Phi_\sigma \wedge \bigwedge_{j\in[m]} C_{j,x}$$

The hash function:
$$h_{\sigma,r_Z}(x) = Z(\Phi_{\sigma,x}, r_Z)$$

Figure 1: The construction of Collision Resistant Hash from any one-way function.

By the compressing properties of Z we get that h_{σ,r_Z} indeed shrinks its input (note that shrinkage by a single bit allows further shrinking by composition). We also have that sampling h_{σ,r_Z} from $\mathcal{H}$ can be done efficiently (with secret coins).

As for collisions, let $x \neq x'$ be two strings in $\{0,1\}^m$ that form a collision, i.e., $h_{\sigma,r_Z}(x) = h_{\sigma,r_Z}(x')$. This equality implies, by the property of the compression, that $\Phi_{\sigma,x}$ is satisfiable iff $\Phi_{\sigma,x'}$ is satisfiable (here we use the fact that the compression is errorless). Due to the binding property of the commitment we have that any assignment satisfying Φ_σ must have $y = i$ (recall that i is the index that σ is a commitment to). Thus the first part of $\Phi_{\sigma,x}$ is only satisfied when $y = i$. But the second part is only satisfied if $x_y = 1$, thus $\Phi_{\sigma,x}$ is satisfied if and only if $x_i = 1$. We get that $\Phi_{\sigma,x}$ is satisfiable if and only if $x_i = 1$ and $\Phi_{\sigma,x'}$ is satisfiable if and only if $x'_i = 1$. Therefore it must be the case that $x_i = x'_i$, since otherwise one of them is 0 and the other one is 1 and $\Phi_{\sigma,x}$ satisfiability is not that of $\Phi_{\sigma,x'}$. necessarily the strings x and x' are such that $x_i = x'_i$. But for some j we have $x_j \neq x'_j$ and for that j we deduce that σ is not a commitment to j.

Suppose now that we have an efficient method of finding a collision x and x' for a given (σ, r_Z). Pick any j such that $x_j \neq x'_j$. Then we know that σ is *not* a commitment to j. This procedure can be used to break the hiding properties of the commitment scheme, since it yields an efficient method that distinguishes the commitment value from random with advantage $1/m$: given (the real) i and a random one $i' \in [m]$ in a random order, run the above procedure to obtain j. If j equals one of the two values i or i', then guess this one as the random one and otherwise flip a coin. This contradicts our assumptions on building blocks (namely, the one-way function).

To prove the result when using compression for any language that is compression-hard for $\mathcal{VC}_{OR}$, a similar construction is defined based on the OR of small circuits rather than CNF formulas: For every $j \in [m]$ let $C_{\sigma,j}$ be the circuit that outputs one if and only if there exists randomness r such that σ is consistent with (j, r) (that is σ is a possible commitment to the value j using randomness r). Let $C_{\sigma,x}$ be the circuit that takes the OR of all $C_{\sigma,j}$ such that $x_j = 1$ and let Z be a compression algorithm for the language OR(CircuitSAT). We define $h_{\sigma,r_Z}(x) = Z(C_{\sigma,x}, r_Z)$. The proof is identical to the case of SAT. □

Note that instead of an errorless compression we can do away with an error probability slightly smaller than 2^{-m}. That is, for all x we want the probability that $Z(\Phi_{\sigma,x}, r_Z)$ preserves the satisfiability of $\Phi_{\sigma,x}$ to be at least $1 - 2^{-m+u}$ where the probability is over σ and r_Z and $u \approx \log m$. In this case we can argue (using a union bound) that with probability at least $1 - 2^{-u}$ no x exists violating the preservation of satisfiability.

We also note that the construction is inherently non-black box as it uses the code of the one-way function (via the commitment) in the application of Cook's Theorem. This is essential for the validity of the whole approach in light of the black-box impossibility of Simon [71]. Theorem 3.1 implies the following corollary:

Corollary 3.2 *If there exists an errorless compression algorithm for SAT or for any problem that is compression-hard for $\mathcal{VC}_{OR}$, then there exist* statistically hiding, computationally binding commitment *schemes based on any one-way function. The scheme requires two rounds of interaction.*

The corollary follows since CRH imply statistically hiding bit commitment, see Naor and Yung [63] (and Damgård, Pedereson and Pfitzman [19] for commitment to many bits). As mentioned in the introduction, the currently known minimal assumptions for constructing statistically hiding bit commitments are the existence of one-way permutations [62] and the more general one-way functions with known pre-image size [38] Furthermore, the commitment schemes of [62, 38] require many rounds of interaction (at least linear in the security parameter), while the commitments based on CRHs are non-interactive, at least after the initial phase where the function $h \in \mathcal{H}$ is chosen. Such a non-interactive CRH also allows for commitment schemes with very low communication [51]. Since the publication of the earlier version of this paper statistically hiding bit commitments based on any one-way function were shown to exist [64, 39]. These protocols, like the previous ones, are based on interactive hashing and therefore require many rounds.

4 On Everlasting Security and the Hybrid Bounded Storage Model

The *bounded storage model*, introduced by Maurer [55], bounds the *space* (memory size) of dishonest players rather than their running time. The model is based on a long random string $\mathcal{R}$ of length m that is publicly transmitted and accessible to all parties. Security relies on the assumption that an adversary cannot possibly store all of the string $\mathcal{R}$ in his memory. The requirement is that the honest parties Alice and Bob can interact using a small local storage (of size n where $n << m$) while security is guaranteed against an eavesdropper Eve with much larger, yet bounded storage space.

This model has enjoyed much success for the task of private key encryption. It has been shown that Alice and Bob who share a short private key can exchange messages secretly using only very small storage[22], while an eavesdropper who can store up to a constant fraction of $\mathcal{R}$ (e.g. $\frac{1}{2}m$ bits) cannot learn anything about the messages (this was shown initially by Aumann and Rabin [5] and improved in [4, 22, 30, 54] and ultimately in Vadhan [73]). These encryption schemes have the important property called *everlasting security* (put forward in [4, 22]). Once the broadcast is over and $\mathcal{R}$ is no longer accessible then the message remains secure even if the private key is exposed and Eve gains larger storage capacity.

In contrast, the situation is less desirable when Alice and Bob do not share any secret information in advance. The solution of Cachin and Maurer [11] for this task requires Alice and Bob to use storage of size at least $n = \Omega(\sqrt{m})$, which is not so appealing in this setting. Dziembowski and Maurer [29] proved that this is also the best one can do.

The Hybrid Bounded Storage Model: The inability to achieve secure encryption in the bounded storage model with memory requirements smaller than $n = \sqrt{m}$ has lead to the following suggestion that we call the *hybrid BSM*: Let Alice and Bob agree on their secret key using a computationally secure key agreement protocol (e.g. the Diffie-Hellman protocol [21]). The rationale being that while an unbounded eavesdropper will eventually break the key, if this happens after the broadcast had already occurred, then the knowledge of the shared key would be useless by then (this should be expected from the everlasting security property where getting the shared key after the broadcast has ended is useless). This hybrid model is very appealing as it attempts to achieve everlasting security by adding assumptions on the ability of an adversary that has a *strict time limit*. Assumptions of this sort are generally very reasonable since all that we require is that the computational protocol is not broken in the short time period between its execution and the transmission of $\mathcal{R}$. For instance, an assumption such as the Diffie Hellman key agreement [21] cannot be broken within half an hour, can be made with far greater degree of trust than actually assuming the long term security of this protocol.

Somewhat surprisingly, Dziembowski and Maurer [29] showed that this rationale may fail. They introduce a specific computationally secure key agreement protocol (containing a non-natural modification based on private information retrieval (PIR) protocols). If this key agreement protocol is used in the hybrid BSM setting with a specific private key scheme, then the eavesdropper can completely decrypt the encrypted message. However, their result does not rule out the possibility that the hybrid idea will work with some other key agreement protocol. For instance, using the plain Diffie Hellman key agreement may still work.

In this work we show that if compression of SAT exists then there exists an attack on the everlasting security of *any* hybrid BSM scheme.

4.1 Two Possible Models

The notation we use for the storage bounds of the honest parties is n_A and n_B (respectively) and for Eve's bound it is m_E. For simplicity we take $n_A = n_B = n$ and use an abuse of notations by setting $m_E = m$ (where actually it should be that $m_E = \frac{1}{2}m$).

We define the hybrid BSM[23] as a setting where the running time of the eavesdropper Eve is polynomially bounded up until and during the broadcast of $\mathcal{R}$, and unbounded after that. We discuss two variants of a

[22] Requires $n = O(\ell + \log m + \log \frac{1}{\varepsilon})$ bits of memory for an ℓ bit message and error ε.

[23] The hybrid BSM model and notions of everlasting security in this model are formally defined in [40].

BSM scheme. We first discuss these in the standard BSM where the eavesdropper is unbounded over time, and then compare them to the hybrid setting where computational restrictions are imposed:

- **The Basic BSM Scheme:** The basic scheme allows interaction only up to the start of the broadcast of $\mathcal{R}$ (after that only the encrypted message is sent). Thus the key is fully determined by the time the broadcast has ended. Such a scheme is fully breakable in the BSM (without an initial secret key) since the unbounded adversary can find some randomness consistent with Alice's view, and simulates Alice's actions and thus recover the encryption key[24]. Basic schemes in the hybrid BSM are interesting as they include any combination of a key agreement protocol with a private key scheme (such as the one described by [29] and [40]). We show that if sufficiently strong compression exists then there exist attacks on any such scheme.

- **The General BSM Scheme:** Alice and Bob interact both before *and* after the broadcast of $\mathcal{R}$. Dziembowski and Maurer [29] show that such a scheme is breakable unless $n^2 > \Omega(m)$ (without initial secret keys). For the hybrid BSM, we show that if compression exists then there exists an attack on any such scheme as long as $n^2 > \Omega(m/p(n, \log m))$, for some polynomial p (related to the polynomial of the compression algorithm and to the running time of the protocol that Alice and Bob use).

Thus we prove that if compression of SAT (or of any $\mathcal{VC}_{OR}$-hard language) is feasible then the hybrid BSM is essentially no more powerful than the standard BSM.

4.2 The Basic Hybrid BSM

Definition 4.1 (Basic hybrid BSM scheme) *A basic hybrid BSM scheme consist of the following: Alice and Bob with storage bound n run a protocol Π that is polynomial in n (this could be a key agreement scheme with security parameter n). Denote by T the transcript of this protocol. Alice and Bob use their respective views of the protocol Π (i.e. the transcript T and their local randomness) to agree on at most n locations of bits from the broadcast string $\mathcal{R}$ that they should store. They store these bits and then use the stored bits to generate an encryption key K (the scheme requires that they agree on the same key).*[25]

We show that sufficiently strong compression of SAT can be used to break any hybrid BSM scheme. For the discussion here take K to be a one bit key. The general idea is that while the eavesdropper may not figure out in time what locations to store, he can use this transcript to save a relatively short (compressed) CNF formula whose satisfiability coincides with the value of the key K. Later, when he is given unbounded computational power, he will be able to extract this bit from the compressed formula.

Theorem 4.2 *If there exists a compression algorithm for SAT or for any compression-hard language for $\mathcal{VC}_{OR}$, with polynomial p_1, then any basic hybrid BSM scheme can be broken using memory $p_2(n, \log m)$ (where p_2 is a polynomial related to p_1 and the running time of the protocol Π).*

Proof: Denote the locations of the bits that Alice and Bob store by $i_1, ..., i_n$. Consider the algorithm V that takes the transcript T_Π and the broadcast string $\mathcal{R}$ as inputs and Alice's local randomness, and locations $i_1, ..., i_n$ as a witness. The algorithm should check if the witness and inputs are indeed consistent with one another (for example, V should verify that a key agreement with the randomness of Alice, the transcript T indeed chooses the indices $i_1, ..., i_n$ to store) and output 1 if and only if they are consistent and generate an

[24]Since Alice must be able to decrypt the message then simulating Alice with any randomness that is consistent with the transcript must output the same key.

[25]The discussion is also valid if the parties are required to reach an agreement with all but negligible probability.

encryption key $K = 1$. The main observation is that the $\mathcal{NP}$ language defined by this relation V is in $\mathcal{VC}_1$. Thus, if SAT has a compression algorithm then there is also a compression algorithm for all of $\mathcal{VC}_1$ (from Lemma 2.17) including the language defined by V.

The attack of the eavesdropper Eve is as follows: Eve generates the verification program V and feeds the instance $(T, \mathcal{R})$ to the compression algorithm for the language V. By the properties of the compression, the output is a CNF formula that is satisfiable if and only if $K = 1$. The length of the output is of some polynomial length $p_2(n, \log m)$. If the polynomial p_2 is sufficiently small then the compressed instance is shorter than Eve's space bound $\frac{1}{2}m$, and he stores this output. Finally, at a later stage, Eve can use her unbounded powers to solve the compressed problem and retrieve the bit K.

We note that a slightly more involved argument works also with compression for $\mathcal{VC}_{OR}$. The idea is to use independent compression for the bit $\mathcal{R}(i_j)$ for every $j \in [n]$. Every such $\mathcal{R}(i_j)$ may be presented as the OR of m circuits of size $p(n)$ each, for some polynomial p. $\Box$

4.3 The General Hybrid BSM

The general scheme is like the basic one but the encryption key K is not necessarily fully defined by the end of the broadcast. In addition, the parties are allowed to interact after the broadcast is over. We note that the bounded storage key exchange scheme of Cachin and Maurer [11] requires such late interaction.

Definition 4.3 (General hybrid BSM scheme) *The general hybrid BSM scheme consist of the following: Alice and Bob with storage bound n engage in a protocol Π_1 that runs in time polynomial in n. Denote by T_1 the transcript of this protocol. Each of the two parties Alice and Bob uses its respective view of the protocol Π_1 to determine at most n locations in the broadcast string $\mathcal{R}$ and stores the bits in these locations. After the broadcast they interact in a second protocol Π_2 (with transcript T_2) at the end of which they both agree on encryption key K (with all but negligible error probability).*

Theorem 4.4 *If there exists compression algorithm for SAT or for any compression-hard language for $\mathcal{VC}_{OR}$ with compression $p_1(n, \log m)$, then there exists an attack on any general hybrid BSM scheme where $n^2 > m/p_2(n, \log m)$ (where p_2 is a polynomial related to p_1 and the running time of the protocol Π_1).*

Proof: Denote by A_{T_1} the set of all possible random strings r_A of Alice that are consistent with the transcript T_1 (recall that T_1 is executed in full before the string $\mathcal{R}$ is broadcast and therefore A_{T_1} is fully determined by T_1). Let $s_A = S_A(T_1, \mathcal{R}, r_A)$ denote the bits that Alice stores at the end of the broadcast when running with randomness r_A, transcript T_1 and broadcast string $\mathcal{R}$. Finally, denote by $\mathbf{S}_A(T_1, \mathcal{R})$ the random variable that is $S_A(T_1, \mathcal{R}, r_A)$ for a uniform choice of $r_A \in A_{T_1}$. That is, $\mathbf{S}_A(T_1, \mathcal{R})$ is distributed over all possible s_A's that Alice may store when running with transcript T_1 and broadcast string $\mathcal{R}$. Similarly we denote by $\mathbf{S}_B(T_1, \mathcal{R})$ the corresponding possible view of Bob.

The proposed strategy for Eve is to store n independent samples from the random variable $\mathbf{S}_A(T_1, \mathcal{R})$. For this purpose we denote by $\mathbf{S}_E(T_1, \mathcal{R})$ (for any $\mathcal{R}$ and T_1) the random variable that consists of n independent samples of $\mathbf{S}_A(T_1, \mathcal{R})$. An important observation due to Maurer [56] is that the uncertainty of Eve regarding the underlying key is upper bounded by the mutual information between the views of Alice and Bob given Eve's view. Formally, the relevant quantity is $I(\mathbf{S}_A(T_1, \mathcal{R}); \mathbf{S}_B(T_1, \mathcal{R}) \mid \mathbf{S}_E(T_1, \mathcal{R}))$. The success of Eve's strategy follows from the two lemmata below, the first due to Dziembowski and Maurer [29] and the second due to Maurer [56]: .

Lemma 4.5 ([29]) *Let $\mathbf{S}_A(T_1, \mathcal{R}), \mathbf{S}_B(T_1, \mathcal{R})$ and $\mathbf{S}_E(T_1, \mathcal{R})$ be defined as above. Then:*

$$I(\mathbf{S}_A(T_1, \mathcal{R}); \mathbf{S}_B(T_1, \mathcal{R}) \mid \mathbf{S}_E(T_1, \mathcal{R})) \leq n^2/m$$

.

Lemma 4.6 ([56], Theorem 3) *Let $\mathbf{V}_A, \mathbf{V}_B$ and $\mathbf{V}_E$ be random variables denoting the respective views of Alice, Bob and Eve. Let $\mathbf{K}_A = K_A(\mathbf{V}_A)$ and $\mathbf{K}_B = K_B(\mathbf{V}_B)$ be procedures of Alice and Bob to extract a mutual secret key from their respective views, such that $\mathbf{K} = \mathbf{K}_A = \mathbf{K}_B$ with all but negligible probability. Then $H(\mathbf{K}) \leq I(\mathbf{V}_A; \mathbf{V}_B \mid \mathbf{V}_E)$.*

A strategy for an eavesdropper is therefore to store n independent samples of the random variable $\mathbf{S}_A(T_1, \mathcal{R})$. Lemmata 4.5 and 4.6 assert that Eve's entropy of the encryption key K is at most n^2/m in such a case. A crucial point is that an encryption key that has entropy significantly lower than 1 (from Eve's point of view) can be predicted with high probability by an unbounded Eve, rendering the scheme insecure. Thus if an eavesdropper has $O(m)$ storage capacity then the scheme is insecure as long as $n^2 < O(m)$.[26]

Lemma 4.5 was used in [29] in a setting where the eavesdropper is unbounded and can hence sample the random variable $\mathbf{S}_A(T_1, \mathcal{R})$. However, in our setting the eavesdropper is computationally bounded and does not have the power to generate this distribution. Instead, we use compression to store information about samples of $\mathbf{S}_A(T_1, \mathcal{R})$ to be extracted *after* the broadcast is over (when the eavesdropper is computationally unbounded).

The main idea is to use compression for search problems, as was discussed in Section 2.7. Define the $\mathcal{NP}$ language L_A as follows:

$$L_A = \{(T_1, \mathcal{R}) | \exists \text{ witness } w = (r_A, s_A) \text{ such that } r_A \in A_{T_1} \text{ and } s_A = S_A(T_1, \mathcal{R}, r_A)\}$$

The first thing to note is that L_A is in $\mathcal{VC}_{OR}$. This is shown once more by the same argument as in Theorems 4.2 or 3.1, and based on the fact that the protocol Π_1 is polynomial time in n. Once this is established, then given a compression algorithm for $\mathcal{VC}_{OR}$ we invoke Theorem 2.25 to get a compression algorithm to the search problem associated with L_A. Running this compression once, allows us to extract a witness to L_A and in particular to get one sample s_A of a consistent view of Alice. Running this n times supposedly gives n samples of such a view, which suffices to break the scheme by Lemma 4.5.

However, in order to invoke Lemma 4.5, we need the samples to be taken according to the distribution $\mathbf{S}_A(T_1, \mathcal{R})$, which is defined by a uniform distribution over $r_A \in A_{T_1}$. We will show that while sampling via the compression of search problems does not give the desired distribution exactly, it is still sufficiently close to be useful.

A closer inspection of our compression for search technique from Section 2.7 shows that we do not necessarily sample uniformly from A_{T_1}. However, we do sample close to uniformly, in the sense that no element in A_{T_1} gets more than double the probability of another element in A_{T_1}. We then show that taking a constant times many samples as was originally needed guarantees that amongst the stored bits we have n random samples of the random variable $\mathbf{S}_A(T_1, \mathcal{R})$, and thus we have stored enough bits from $\mathcal{R}$ to break the scheme.

Recall from Section 2.7 that the compression algorithm for search problems chooses a random pairwise-independent hash function h and saves only a witness (r_A, s_A) that is *uniquely* hashed to the value 0 by h. Since r_A fully determines s_A (when given T_1 and $\mathcal{R}$), then without loss of generality we view the witness simply as r_A. Furthermore, assume w.l.o.g. that r_A is of length n. Suppose that $\ell \in [n]$ is such that $2^\ell < |A_{T_1}| \leq 2^{\ell+1}$. Let $\mathcal{H}_{\ell+2}$ be a family of pairwise independent hash functions with $h : \{0,1\}^n \to \{0,1\}^{\ell+2}$ for all $h \in \mathcal{H}_{\ell+2}$. Then for every $r_A \in A_{T_1}$ the probability that a random $h \in \mathcal{H}_{\ell+2}$ uniquely maps r_A to zero is at most $2^{-(\ell+2)}$ (since $\Pr_{h \in \mathcal{H}_{\ell+2}}[h(r_A) = 0] = 2^{-(\ell+2)}$). By the pairwise independence of $\mathcal{H}$ it holds that for all other $r'_A \in A_{T_1}$ with $r'_A \neq r_A$ we have that $\Pr_{h \in \mathcal{H}_{\ell+2}}[h(r'_A) \neq 0 | h(r_A) = 0] = 1 - 2^{-(\ell+2)}$. By

[26] When considering n_A and n_B that are not necessarily identical, the actual requirement is for Eve to store n_B samples of $\mathbf{S}_A(T_1, \mathcal{R})$ (each sample is of length n_A). Subsequently the scheme is insecure as long as $n_A \cdot n_B < O(m_E)$.

a union bound over all $r'_A \in A_{T_1}$ with $r'_A \neq r_A$, combined with the probability that $h(r_A) = 0$, we get:

$$\Pr_{h \in \mathcal{H}_{\ell+2}}[h \text{ uniquely maps } r_A \text{ to } 0] \geq 2^{-(\ell+2)} \cdot \frac{1}{2} = 2^{-(\ell+3)}.$$

Altogether, for all $r_A \in A_{T_1}$ it holds that

$$2^{-(\ell+2)} \geq \Pr_{h \in \mathcal{H}_{\ell+2}}[h \text{ uniquely maps } r_A \text{ to } 0] \geq 2^{-(\ell+3)}.$$

Thus whenever the hash used is indeed of length $\ell + 2$, the probability of sampling $r_A \in A_{T_1}$ is almost uniform (up to a factor of 2 for each element).[27] is no Since we repeat the compression for every choice of $\ell \in [n]$, then in particular samples are stored for the correct ℓ.

By Lemma 2.26 we know that at least $\frac{1}{8}$ of the repeated compressions indeed store information about a valid witness (a sample of $r_A \in A_{T_1}$). Thus, choosing, say, $9n$ independent $h \in \mathcal{H}_{\ell+2}$ guarantees at least n samples (by a Chernoff bound, as the choices are independent). But as mentioned above, these samples are just close to uniform over A_{T_1} rather than truly uniform. The solution is to simply run more instances of this process, say, for $25n$ independent choices of $h \in \mathcal{H}_{\ell+2}$. This would guarantee that with overwhelming probability, at least $3n$ of these choices have a valid witness. We show that from these slightly biased samples we can extract n truly uniform samples of witnesses.

This last argument follows by a method for generating uniformly distributed samples from A_{T_1}. At a first stage, $3n$ samples are taken using the unique hashing method. Now a diluting second stage is in order run to extract the actual samples: Suppose that the least likely element to be sampled gets probability p_{min}. For any element r_A that is sampled with probability p_{r_A}, keep the sample with probability $\frac{p_{min}}{p_{r_A}}$ and delete it otherwise. Thus every element is eventually chosen with the same probability p_{min}, and since $\frac{p_{min}}{p_{r_A}} \geq \frac{1}{2}$ then at least n samples are eventually chosen (with overwhelming probability). Note that the diluting stage is not necessarily efficiently computable. However, the probability p_{r_A} can be computed using the adversaries unbounded running time, since these probabilities are fully defined by the transcript T_1 which can be stored in its entirety (as it is of length polynomial in n). Therefore an unbounded eavesdropper may indeed extract n uniform samples from her view. □

Note: In the two models that we consider we limit the honest parties to access and store at most n actual bits from the broadcast string $\mathcal{R}$. This is in contrast to storing some function of $\mathcal{R}$ with a bound on the function's output length (an ability that the adversary is entitled to). This is a legitimate requirement as the honest parties should run algorithms that are considerably more efficient than the adversary's. It should be noted, however, that our Theorems (4.2 and 4.4) hold also if the honest players can store functions, albeit they then call for a compression algorithm for all of $\mathcal{NP}$ (rather than just for the lowest class $\mathcal{VC}_{OR}$).

5 On Witness Retrievable Compression and Public Key Cryptography Based on Any One-Way Function

5.1 On Oblivious Transfer from any One-Way Function

As mentioned in the introduction, whether one-way functions are sufficient for public key cryptography is a long standing open problem. In fact, many researchers view the black-box impossibility result of Impagliazzo and Rudich [46] as an indication that general one-way functions are insufficient for public key cryptography. We now describe an approach to bridging this gap using witness retrievable compression of a specific language.

[27] Note that the almost uniformity of the samples actually holds for every choice of the parameter ℓ. Therefore, this property can be relied on even if the correct choice of ℓ is unknown.

Theorem 5.1 *If there exists a witness retrievable compression algorithm for a specific type of SAT formulas, then there exists an Oblivious Transfer (OT) protocol based on any one-way function.*

Proof: The construction actually builds a Private Information Retrieval (PIR) protocol, and then uses the construction of Di Crescenzo, Malkin and Ostrovsky [20] to build an OT protocol from the PIR protocol. Recall that a PIR protocol has a sender with a database of size m and a receiver that chooses to learn one entry from the database. It is required that the receiver learns the bit of his choice, but a computationally bounded sender learns essentially nothing about this choice. In addition, the total communication should be strictly smaller than m.

Let f be a one-way function and take $(\mathrm{C}ommit, \mathrm{V}erify)$ to be a commitment based on the one-way function f (as in Section 3). In this proof we work under the assumption that the parties are semi-honest (that is, the parties follow the protocol as prescribed and are only allowed to try and infer extra information from the transcript of the protocol). The semi-honest assumption is justified by the compiler of Goldreich, Micali and Wigderson [35] that showed how to transform a semi-honest protocol into one against malicious parties (again, the only needed cryptographic assumption is the existence of a one-way function). Consider the protocol in Figure 2.

Protocol PIR_f:
Alice's input: database D of m bits. Let $D[i]$ denote the ith bit in D.
Bob's input: index $i \in [m]$ denote the bits of i by $i_1, ..., i_\ell$

1. **Bob commits to i:** Bob commits to i with randomness r_B, Alice receives $\sigma = \mathrm{C}ommit(i, r_B)$.
2. **Alice computes Φ:** The CNF formula Φ is defined as follows:
   - Denote by $\mathrm{V}erify_\sigma$ the algorithm $\mathrm{V}erify$ with the input σ fixed. That is, $\mathrm{V}erify_\sigma$ takes as inputs x and r and accepts if and only if they form a legal opening of the commitment σ (and in particular this means that $x = i$).
   - Translate $\mathrm{V}erify_\sigma$ into a CNF formula Φ_σ over the variables $x_1, ..., x_\ell$ of x and the bits of r (using Cook's reduction).
   - For every $j \in [m]$ define the clause $C_j = (x_1^{\bar{j}_1} \vee x_2^{\bar{j}_2} \vee .... \vee x_\ell^{\bar{j}_\ell})$ if $D[j] = 0$ (where x^0 denotes $\bar{x}$ and x^1 denotes x) and $C_j = 1$ if $D[j] = 1$.
   - Set
   $$\Phi = \Phi_\sigma \wedge \bigwedge_{j \in [m]} C_j$$
3. **Alice Compresses Φ:** Let (Z, W) be a witness retrievable compression algorithm for CNF formulas of the form of Φ. Alice runs $\Psi = Z(\Phi)$ and sends Ψ to Bob.
4. **Bob checks witness:** Note that Bob knows the witness to $\mathrm{V}erify_\sigma$ and can compute a witness w for Φ_σ. Bob checks if $W(w, \Psi)$ is a satisfying assignment for Ψ. If it is Bob outputs 1, otherwise he outputs 0.

Figure 2: The construction of a PIR protocol from any one-way function.

It remains to show that the protocol PIR_f is indeed a PIR protocol. Due to the fact that the commitment is binding (up to a negligible error), then an assignment satisfying Φ_σ must have $x = i$ (recall that i is the index that Bob committed to). Thus the first part of Φ is only satisfied when $x = i$. But the second part is only satisfied if $D[x] = 1$, thus Φ is satisfied if and only if $D[i] = 1$. By the property of the compression algorithm, also Ψ is satisfiable iff $D[i] = 1$. Hence, using the witness retrievable properties of the compression, Bob figures out whether or not Ψ is satisfiable, and learns the bit $D[i]$ (up to a negligible error).

The second property is that the sender Alice learns no computational information about Bob's choice. This follows directly from the guarantees of the commitment scheme (note that Bob does not send any information outside of the commitment). The third and final requirement regards the length of the communication. But the length of the communication is a fixed polynomial in $p(n)$ (depending on the commitment protocol and the parameter of the compression algorithm). So choosing a large enough databases with $m > p(n)$ guarantees a non trivial PIR protocol and hence an OT protocol. $\square$

Note that the OT protocol derived in Theorem 5.1 is a one-round protocol (that is, one message sent from the receiver followed by one message from the sender). This follows from the construction of the PIR protocol and the construction of [20] that preserves the number of rounds. One implication of this fact is that such an OT protocol may be used to construct a two round key agreement scheme, that in turn maybe used to construct a public key encryption. In general, this is achieved by fixing the first message of the protocol to be as the public key. Formally:

Corollary 5.2 *If there exists a witness retrievable compression algorithm for a specific type of SAT instances, then based on any one-way function one can construct a public key encryption scheme (PKE) that is semantically secure against chosen plaintext attacks.*

5.2 On the Limitation of the Witness Retrievability Property

Witness retrievable compression is defined (Definition 1.5) as a compression with an additional algorithm W such that for every witness w_x for R_L it holds that $w_y = W(w_x, Z(x))$ is a witness for $Z(x) \in L'$. Recall that all of the examples of compression algorithms (in Sections 2.1 and 2.9) are in fact witness retrievable. This property is essential to the success of the construction of the OT protocol in Theorem 5.1, (without it the receiver would have to run in time that is super-polynomial). In this section we show that if one-way functions exist then a compression algorithm for SAT cannot be witness retrievable (this regards the general language SAT rather than a specific distribution of instances as generated in Theorem 5.1). Moreover, this statement also holds for other general languages mentioned in Theorem 5.1 (that are potentially easier to compress than SAT). In particular, there is no witness retrievable compression for the Clique language or for the language $OR(SAT)$ (that is complete for $\mathcal{VC}_{OR}$). We give the formal statements below with respect to the language $OR(SAT)$ and deduce the statements for SAT and Clique as corollaries.

We also rule out other natural definitions of witness retrievability that would have been sufficient for the proof of Theorem 5.1 to go through. Suppose we relax the witness retrievability requirement to hold only with some probability ε, then we show that if one-way functions exist then this probability ε has to be very low, at most an inverse polynomial in m. Such a low probability of success is *not* sufficient for the OT construction in Theorem 5.1 to follow (we note though, that witness retrievability with this low success probability is still sufficient for the cryptanalytic result in [28]). We then show that the same situation also holds for languages that are guaranteed to have *unique witnesses* (i.e. unique-SAT and unique-OR(SAT)). This is of relevance since the instances being compressed in the proof of Theorem 5.1 all have at most a single witness.[28]

We emphasize that the OT construction may still be successful under the compression of formulas of the specific type that are generated in the proof. However, we cannot generalize this method to work with compression of a more standard language.

On the Impossibility of *Perfect* Witness Retrieval: Recall that the language $OR(SAT)$ takes as an input a list of m CNF formulas (each of length n) and accepts if at least one of the formulas is satisfiable. Consider the following way of generating an instance of $OR(SAT)$. Take m bit commitments $\sigma_1, \ldots, \sigma_m$, each with security parameter n. For each commitment σ_i, generate using Cook's Theorem a CNF formula ϕ_{σ_i} that is satisfiable if and only if σ_i is a commitment to 1. As an instance of $OR(SAT)$ we take the OR of the m CNF formulas $\phi_{\sigma_1}, ..., \phi_{\sigma_m}$. We denote this instance by $\phi(\sigma_1, \ldots, \sigma_m)$. Denote by w_{σ_i} a satisfying assignment for ϕ_{σ_i} (such an assignment can be generated by an opening σ_i to the value 1). The assignment w_{σ_i} also serves as a witness for $\phi(\sigma_1, \ldots, \sigma_m) \in OR(SAT)$. Our first impossibility result is for compression of $OR(SAT)$ with errorless witness retrievability.

Lemma 5.3 *If one-way functions exist then there is no witness retrievable compression for $OR(SAT)$ with perfect witness retrieval.*

Proof: The proof follows by showing that a witness retrievable compression Z for $OR(SAT)$ can be used to transmit an m bit string between two parties with sub-linear communication. As a setup stage, the receiver generates m random commitments to 1 and m random commitments to 0 and sends them to the sender. Denoted these by $(\sigma_1^1, \ldots, \sigma_m^1)$ and $(\sigma_1^0, \ldots, \sigma_m^0)$ respectively.

For every string $x \in \{0,1\}^m$ denote $\phi_x = \phi(\sigma_1^{x_1}, \ldots, \sigma_m^{x_m})$ (where x_i denotes the i^{th} bit of x). In order to send string $x \in \{0,1\}^m$ the sender sends $Z(\phi_x)$ to the receiver. We claim that the receiver can, with overwhelming probability, learn the string x, thus contradicting the fact that the message sent is significantly shorter than m. Note that the receiver knows witnesses $w_{\sigma_i^1}$ for all i and that a witness for $\phi_x \in OR(SAT)$ consists of a witness $w_{\sigma_i^1}$ of a $\phi_{\sigma_i^1}$ that is included in ϕ_x. The receiver extracts x as follows:

Procedure Rec on input $Z(\phi_x)$:

- For every $i \in [m]$:
  1. Run $W = W(Z(\phi_x), w_{\sigma_i^1})$
  2. If W is a witness for $Z(\phi_x)$ then set $y_i = 1$, otherwise, set $y_i = 0$.
- Output $y = y_1, ..., y_m$.

Denote by X_i the random variable of the i^{th} bit of x and by Y_i the random variable of the corresponding output of Rec. We view the process as a channel between a sender who holds the random variables $X = X_1, ..., X_m$ to a receiver who gets the random variables $Y = Y_1, ..., Y_m$ and claim that with overwhelming probability $Y = X$.

If $X_i = 1$ then the opening of σ_i^1 should yield a witness for $Z(\phi_x)$, from the perfect witness retrievability, and thus $Y_i = 1$. We should show that if $X_i = 0$, then indeed $Y_i = 0$ (up to a negligible error). Note that X is uniformly distributed over $\{0,1\}^m$, whereas Y is determined by the random choice of commitments $(\sigma_1^1, \ldots, \sigma_m^1)$ and $(\sigma_1^0, \ldots, \sigma_m^0)$, the random coins of Z and W and the random variable X.

[28] The relevant instances in Theorem 5.1 actually have a unique witness only if there exists a commitment scheme that has only a *unique opening*. As this is not necessarily the case when given any one-way function, we consider for simplicity the case of one-way permutations (that guarantee a unique opening commitment scheme).

Claim 5.4 *Let X and Y be the random variables described above. Then for every $i \in [m]$ and every polynomial $q(\cdot)$,*

$$\Pr[Y_i = 1 | X_i = 0] < \frac{1}{q(n)}.$$

Proof: Suppose that the claim is false, that is, for some $q(\cdot)$ and some i, $\Pr[Y_i = 1|X_i = 0] \geq 1/q(n)$. For simplicity we first deal with the case that $\Pr[Y_i = 1|X_i = 0] = 1$. In other words, $W(Z(\phi_x), w_{\sigma_i^1})$ always outputs a witness for $Z(\phi_x)$. Consider the two distributions $\mathcal{L}_0$ and $\mathcal{L}_1$ on lists of $m-1$ commitments:

- Distribution $\mathcal{L}_0$ is defined by a random and independent choice of $m-1$ commitments to 0.
- Distribution $\mathcal{L}_1$ is defined by first choosing at random a string $V_1, V_2, \ldots, V_{m-1} \in \{0,1\}^{m-1}$ and then generating $m-1$ independent commitments to $V_1, V_2, \ldots, V_{m-1}$.

¿From the hiding property of commitment schemes it holds that these two distributions are indistinguishable, i.e. given a list L of $m-1$ commitments, no computationally bounded distinguisher can tell with non-negligible bias whether L was generated by $\mathcal{L}_0$ or $\mathcal{L}_1$. We will show that if the premise of the claim is false, it is possible to distinguish the two distributions (without knowledge of the openings to any of the commitments in the list).

Given a list L of $m-1$ commitments, the distinguisher generates σ_i^0 and σ_i^1 and the corresponding witnesses. He then generates a formula ϕ by adding σ_i^0 to the i^{th} position in the list L, and runs the compression on ϕ. The distinguisher then runs $W = W(Z(\phi), w_{\sigma_i^1})$ and checks whether W is a witness to $Z(\phi)$. By the assumption, W will indeed be a witness every time that ϕ is satisfiable. On the other hand, W cannot be a witness if ϕ is *not* satisfiable, simply by the properties of the compression. Thus if W is indeed a witness for $Z(\phi)$ then it must be that $\phi \in OR(SAT)$ and there is some commitment to 1 in the list and thus L was generated from $\mathcal{L}_1$. Otherwise, it means that $\phi \notin OR(SAT)$ and the original list was from $\mathcal{L}_0$ (ignoring the negligible probability that $\mathcal{L}_1$ generates a list containing only commitments to 0).

Now if $\Pr[Y_i = 1|X_i = 0] \geq \frac{1}{q(n)}$ for some polynomial $q(\cdot)$, then the distinguisher follows the same procedure with the difference that:

- If $W = W(Z(\phi), w_{\sigma_i^1})$ is a witness for $Z(\phi)$ then output $\mathcal{L}_1$.
- If W is not a witness flip a coin and answer either $\mathcal{L}_0$ or $\mathcal{L}_1$ accordingly.

In case W was indeed a witness, the distinguisher is guaranteed to be correct. Therefore, the above procedure gives an advantage $\frac{1}{2q(n)}$ in distinguishing between $\mathcal{L}_0$ and $\mathcal{L}_1$, contradicting the hiding properties of the commitment scheme. □

Note that the distributions $\mathcal{L}_0$ and $\mathcal{L}_1$ will be useful also in the discussion of the unique witnesses case (Lemma 5.6). □

On *Non-Perfect* Witness Retrievability: We now show that the witness retrieval procedure is possible only if its success probability is sufficiently low (we denote the success probability by $\frac{1}{q(n,m)}$). We upper bound the success probability by a function of the rate of compression that the algorithm Z achieves (we denote by $p(n,m)$ the polynomial that bounds the length of the output of Z, i.e. the compressed instance).

Lemma 5.5 *If one-way functions exist and suppose that (Z, W) is a witness retrievable compression for $OR(SAT)$ such that for every ϕ with parameters m, n the following holds:*

1. *The compression parameter $|Z(\phi)| \leq p(n,m)$*

2. *The success probability of W is at least $\frac{1}{q(n,m)}$ where probability is over the random coins of Z and W as well as the choice of the witness.*

Then $q(n,m) \geq \Omega(\frac{m}{p(n,m)})$.

Proof: The proof uses the same setting as in the proof of Lemma 5.3. Once more, the sender sends a compressed value $Z(\phi_x)$ to the receiver that runs the procedure Rec and we view this process as a channel between a sender who holds the random variables $X = X_1, ..., X_m$ to a receiver who gets the random variables $Y = Y_1, ..., Y_m$. Only this time if $X_i = 1$ it is not guaranteed that also $Y_i = 1$ (since the witness retrievability is no longer perfect). Instead, our assumption on the success probability of W translates to $\Pr[Y_i = 1 \mid X_i = 1] \geq \frac{1}{q(n,m)}$ for a random i. Since X_i is a uniformly distributed bit then $\Pr[Y_i = 1] \geq \frac{1}{2q(n,m)}$ for a random i.

In addition, Claim 5.4 states that for every i it holds that $\Pr[Y_i = 1 \mid X_i = 0] \in neg(n)$. Thus, if $Y_i = 1$ then $X_i = 1$ with overwhelming probability and therefore $H(X_i \mid Y_i = 1) \in neg(n)$ for every i (where H denotes the Shannon entropy). We use the above mentioned facts to provide an upper bound on the average entropy of X_i (average over i) when given Y:

$$\begin{aligned}
\mathbb{E}_i[H(X_i \mid Y)] &= \mathbb{E}_i[Pr(Y_i = 1)H(X_i \mid Y_i = 1) + Pr(Y_i = 0)H(X_i \mid Y_i = 0)] \\
&\leq \frac{1}{2q(n,m)} \cdot neg(n) + (1 - \frac{1}{2q(n,m)}) \cdot 1 \\
&\leq 1 - \frac{1}{2q(n,m)} + neg(n)
\end{aligned}$$

The last inequality is true since $H(X_i \mid Y_i = 0) \leq 1$ for every i. We deduce an upper bound on the entropy of X when given Y:

$$H(X|Y) \leq \sum_i H(X_i \mid Y) = m\mathbb{E}_i[H(X_i \mid Y)] \leq m(1 - \frac{1}{2q(n,m)} + neg(n))$$

Hence, when the receiver gets $Z(\phi_x)$ (and can generate Y), the receiver's entropy of X deteriorates by

$$H(X) - H(X \mid Y) \geq \Omega(\frac{m}{q(n,m)}).$$

This can only happen if the sender sent at least $\Omega(\frac{m}{q(n,m)})$ bits to the receiver, and thus $p(n,m) \geq \Omega(\frac{m}{q(n,m)})$ as required. □

Note that the construction of OT protocols from one-way functions in Theorem 5.1 requires that the compression rate $p(n,m) \leq O(m^{1-\epsilon})$ for some constant $\epsilon > 0$. Thus, when put in the context of constructing OT protocols, the above lemma states that a useful compression algorithm for $OR(SAT)$ cannot have witness retrievability with probability that is better than $O(\frac{1}{m^\epsilon})$.

On Witness Retrieval with a *Unique Witness*: The limitations on witness retrievability hold also when there is only a single witness, which is the case in our cryptographic applications. For this we consider the *promise problem* $OR(SAT)^U$ that is $OR(SAT)$ with a guarantee that every instance has at most one satisfying assignment. We generate the interesting instances of $OR(SAT)^U$ as above, from sets of commitments. In this case the set of commitments should be such that at most one of the commitments is to the value 1. For simplicity we also assume that each commitment has a unique opening (this may be achieved using one-way permutation), so overall such instances have the unique witness property.

Lemma 5.6 *If one-way permutations exist and suppose that (Z, W) is a witness retrievable compression for $OR(SAT)^U$ such that for every input ϕ with parameters m, n the following holds:*

1. *The compression parameter $|Z(\phi)| \leq p(n, m)$*
2. *The success probability of W is at least $\frac{1}{q(n,m)}$ for a polynomial $q(\cdot, \cdot)$ where probability is over the random coins of Z and W.*

Then $\frac{1}{q(n,m)} - \frac{p(n,m)}{m} \in neg(n)$.

Proof: Suppose that there is a witness retrievable compression (Z, W) for $OR(SAT)^U$ that succeeds with probability $\frac{1}{q(n,m)}$. In similar fashion to the proof of Claim 5.4 we will show that in such a case one can efficiently distinguish if a list of $m - 1$ commitments was generated by the distribution $\mathcal{L}_0$ or by the distribution $\mathcal{L}_1$. Recall that the distribution $\mathcal{L}_0$ is a random choice of $m - 1$ commitments to 0 while the distribution $\mathcal{L}_1$ is a choice of $m - 1$ random commitments (commitments to either 0 or 1). The distinguisher works without knowledge of the openings to any of the commitments, thus contradicting the hiding properties of the commitment scheme.

The distinguisher generates a random commitment σ^1 to 1 along with its witness w_{σ^1}. Now, given a list L of $m - 1$ commitments, the distinguisher creates an instance ϕ by adding σ^1 in a random position in the list L, and runs the compression on ϕ. The distinguisher then tries to retrieve a witness to $Z(\phi)$ using the opening w_{σ^1}. In the case that $L \in \mathcal{L}_0$ then ϕ is an instance of $OR(SAT)^U$ and thus by the assumption the distinguisher will retrieve a witness with probability at least $\frac{1}{q(n,m)}$. On the other hand, if $L \in \mathcal{L}_1$ then the instance ϕ is a general instance of $OR(SAT)$ (without the promise of the unique witness). Lemma 5.5 states that there exists a ϕ for which the witness retrieval succeeds with probability at most $\frac{p(n,m)}{m}$. A more careful inspection of the proof of Lemma 5.5 shows that this statement also holds for a randomly chosen ϕ (generated by choosing m random commitments not all of which are to 0). Thus, if $L \in \mathcal{L}_1$ then the witness retrieval succeeds on ϕ with probability at most $\frac{p(n,m)}{m}$ (with probability taken over the choice of $L \in \mathcal{L}_1$ and the randomness of the distinguisher). Overall, the distinguisher accepts with probability at least $\frac{1}{q(n,m)}$ when L is from $\mathcal{L}_0$ and at most $\frac{p(n,m)}{m}$ when L is from $\mathcal{L}_1$. So if $\frac{1}{q(n,m)} - \frac{p(n,m)}{m}$ is larger than a polynomial fraction in n, then this procedure has a distinguishing advantage between $\mathcal{L}_0$ and $\mathcal{L}_1$, contradicting the security of the commitment scheme. $\Box$

All our results have been stated for the language $OR(SAT)$. However, they may be applied for other languages such as SAT and Clique. In particular, we get the statement with respect to SAT as a corollary (since a compression for SAT can be used as a compression for $OR(SAT)$ via the reduction in Lemma 2.17).

Corollary 5.7 *If one-way functions exist and let (Z, W) be a witness retrievable compression for SAT (or for Unique-SAT), such that for every input ϕ with parameters m, n the following holds:*

1. *The compression parameter $|Z(\phi)| \leq p(n, m)$*
2. *The success probability of W is at least $\frac{1}{q(n,m)}$ where probability is over the random coins of Z and W as well as the choice of the witness.*

Then $q(n, m) \geq \Omega(\frac{m}{p(n,m)})$.

6 Discussion and Open Problems

The issue of compressibility and the corresponding classification introduced in this work raise many open problems and directions. The obvious one is to come up with a compression algorithm for a problem like SAT or Clique (or some $\mathcal{VC}_{OR}$ complete or hard problem). Alternatively, show why such tasks are infeasible (see discussion in Section 2.9). We have seen compressibility of some interesting $\mathcal{NP}$ languages and hence the question is where exactly is boundary between compressibility and incompressibility. We tend to conjecture that it is in the low levels of the $\mathcal{VC}$ hierarchy. We view PCP amplification methods such as the recent result of Dinur [23] as potential leads towards achieving compression. This is since these results show a natural amplification of properties on a graph, and could potentially be combined with a simple compression of promise problems (such as the example for GapSAT in Section 2.9). The main issue is doing the PCP amplification without introducing many new variables.

In particular, the following task would suffice for achieving non-trivial compression: given CNF formulae ϕ_1 and ϕ_2 (not necessarily with short witnesses) come up with a CNF formula ϕ that is (1) satisfiable if and only if $\phi_1 \vee \phi_2$ is satisfiable and (2) shorter than the combined lengths of ϕ_1 and ϕ_2 ; By shorter we mean of length $(1-\epsilon)(|\phi_1| + |\phi_2|)$. The reason this is sufficient is that we can apply it recursively and obtain non-trivial compression for OR(SAT), which implies the cryptographic applications. Moreover, due to the impossibility results for general witness retrievable compression (Section 5.2), a witness for either ϕ_1 or ϕ_2 cannot efficiently yield a witness for ϕ.

Short of showing a compression for general complexity classes, it would be interesting to come up with further interesting compression algorithms as well as to obtain more hardness results. For instance, is Clique or any other embedding problem complete for $\mathcal{VC}_1$? Is there a natural and simple complete problem for $\mathcal{VC}_1$? Also, the $\mathcal{VC}$ hierarchy is by no means the ultimate classification with respect to compressibility. One can hope to further refine this classification, especially within the confines of $\mathcal{VC}_1$. Moreover, it would be interesting to find connections of the $\mathcal{VC}$ hierarchy to other classifications (e.g., in the style of Feige [31] for average case complexity and approximation algorithms and Chen et al. [14] for parameterized complexity and subexponential algorithms).

Regarding the cryptographic application of getting a CRH from one-way functions (Theorem 3.1), one issue is how essential is the requirement that the compression will be errorless. We know that this requirement can be relaxed to hold with an error that is exponentially small in m. However it is unknown whether a CRH can be constructed from any one-way function using a compression algorithm that errs with probability that is, say, exponentially small in n and $\log m$. Note that using typical amplification techniques for CRH is unsuccessful. For example, taking a concatenation of several independently chosen hash functions on the same input fails, since reducing the adversary's success probability to allow using the a union bound requires using too many ($\Omega(m)$) independent functions for the overall hash to still be shrinking.

Since we currently do not have a general compressibility result for a significant class of languages, it is important to understand what are the implications of *incompressibility*. The application to the bounded storage model can be viewed as such a statement. Other examples are the previously mentioned works of Dubrov and Ishai [26] regarding derandomization and Dziembowski [28] with respect to forward-secure storage. In order to gain confidence in an incompressibility assumption when used in a cryptographic setting it is important to come up with an *efficiently falsifiable* assumption[29] of this nature (see [61]).

Finally we feel that we have just scratched the surface of an important topic and in the future there will be other implications of compressibility or the impossibility of compression, whether in cryptography or in other areas.

[29] An efficiently falsifiable assumption is one for which it is possible to create verifiable challenges so that if the assumption is false then the challenges can be solved.

Acknowledgements: We thank Yuval Ishai for many helpful comments and specifically for pointing out that the CRH construction does not require witness retrievability. We are also grateful to Alon Rosen, Ronen Shaltiel and Gillat Kol for their comments on the presentation and Hovav Shacham for conversations regarding witness retrievable compression. Finally we thank the anonymous FOCS referees and Mike Langston for their helpful comments and suggestions and to Mike Fellows for pointing out some references.

References

[1] S. Aaronson. NP-complete problems and physical reality. *SIGACT News*, 36(1):30–52, 2005.

[2] M. Abadi, J. Feigenbaum, and J. Kilian. On hiding information from an oracle. *Journal of Computer and System Sciences (JCSS)*, 39(1):21–50, 1989.

[3] N. Alon, R. Yuster, and U. Zwick. Color-coding. *Journal of the ACM*, 42(4):844–856, 1995.

[4] Y. Aumann, Y.Z. Ding, and M.O. Rabin. Everlasting security in the bounded storage model. *IEEE Transactions on Information Theory*, 48(6):1668–1680, 2002.

[5] Y. Aumann and M.O. Rabin. Information theoretically secure communication in the limited storage space model. In *Advances in Cryptology – CRYPTO '99, Lecture Notes in Computer Science*, volume 1666, pages 65–79. Springer, 1999.

[6] B. Barak. How to go beyond the black-box simulation barrier. In *42nd IEEE Symposium on Foundations of Computer Science*, pages 106–115, 2001.

[7] M. Bellare, A. Boldyreva, and A. Palacio. An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In *Advances in Cryptology – EUROCRYPT ' 2004, Lecture Notes in Computer Science*, volume 3027, pages 171–188. Springer, 2004.

[8] S. Ben-David, B. Chor, O. Goldreich, and M. Luby. On the theory of average case complexity. *Journal of Computer and System Sciences (JCSS)*, 44(2):193–219, 1992.

[9] K. Burrage, V. Estivill-Castro, M. Fellows, M. Langston, S. Mac, and F. Rosamond. The undirected feedback vertex set problem has a poly() kernel. In *Parameterized and Exact Computation, Second International Workshop (IWPEC 2006), Lecture Notes in Computer Science*, volume 4169, pages 192–202. Springer, 2006.

[10] J. Buss and T. Islam. Simplifying the weft hierarchy. *Theoretical Computer Science*, 351(3):303–313, 2006.

[11] C. Cachin and U. Maurer. Unconditional security against memory-bound adversaries. In *Advances in Cryptology – CRYPTO '97, Lecture Notes in Computer Science*, volume 1294, pages 292–306. Springer, 1997.

[12] L. Cai and J. Chen. On the amount of nondeterminism and the power of verifying. *SIAM Journal of Computing*, 26(3):733–750, 1997.

[13] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. *Journal of the ACM*, 51(4):557–594, 2004.

[14] J. Chen, B. Chor, M. Fellows, X. Huang, D. Juedes, I. Kanj, and G. Xia. Tight lower bounds for certain parameterized NP-hard problems. *Information and Computation*, 201(2):216–231, 2005.

[15] J. Chen, I. Kanj, and W. Jia. Vertex cover: Further observations and further improvements. *Journal of Algorithms*, 41(2):280–301, 2001.

[16] B. Chor, M. Fellows, and D. Juedes. Linear kernels in linear time, or how to save k colors in $O(n^2)$ steps. In *WG 04, Lecture Notes in Computer Science*, volume 3353, pages 257–269. Springer-Verlag, 2004.

[17] S.A. Cook. The complexity of theorem-proving procedures. In *3rd ACM Symposium on the Theory of Computing*, pages 151–158, 1971.

[18] I. Damgård. A design principle for hash functions. In *Advances in Cryptology - CRYPTO '89, Lecture Notes in Computer Science*, volume 435, pages 416–427. Springer, 1989.

[19] I. Damgård, T. Pedersen, and B. Pfitzmann. On the existence of statistically hiding bit commitment schemes and fail-stop signatures. In *Advances in Cryptology - CRYPTO '93, Lecture Notes in Computer Science*, volume 773, pages 250–265. Springer, 1993.

[20] G. Di Crescenzo, T. Malkin, and R. Ostrovsky. Single database private information retrieval implies oblivious transfer. In *Advances in Cryptology – EUROCRYPT ' 2000, Lecture Notes in Computer Science*, volume 1807, pages 122–138. Springer, 2000.

[21] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Transaction on Information Theory*, 22:644–654, 1976.

[22] Y.Z. Ding and M.O. Rabin. Hyper-encryption and everlasting security. In *Annual Symposium on Theoretical Aspects of Computer Science (STACS), Lecture Notes in Computer Science*, volume 2285, pages 1–26, 2002.

[23] I. Dinur. The PCP theorem by gap amplification. In *38th ACM Symposium on the Theory of Computing*, pages 241–250, 2006.

[24] R. Downey and M. Fellows. **Parameterized Complexity**. Springer-Verlag, 1999.

[25] R. Downey, M. Fellows, and U. Stege. Parameterized complexity: a systematic method for confronting computational intractability. In *Contemporary Trends in Discrete Mathematics, AMS DIMACS Proceedings Series*, volume 49, pages 49–100, 1999.

[26] B. Dubrov and Y. Ishai. On the randomness complexity of efficient sampling. In *38th ACM Symposium on the Theory of Computing*, pages 711–720, 2006.

[27] C. Dwork, J. Lotspiech, and M. Naor. Digital signets: Self-enforcing protection of digital information. In *28th ACM Symposium on the Theory of Computing*, pages 489–498, 1996.

[28] S. Dziembowski. On forward-secure storage. In *Advances in Cryptology – CRYPTO 2006, Lecture Notes in Computer Science*, volume 4117, pages 251–270. Springer, 2006.

[29] S. Dziembowski and U. Maurer. On generating the initial key in the bounded-storage model. In *Advances in Cryptology – EUROCRYPT ' 2004, Lecture Notes in Computer Science*, volume 3027, pages 126–137. Springer, 2004.

[30] S. Dziembowski and U. Maurer. Optimal randomizer efficiency in the bounded-storage model. *Journal of Cryptology*, 17(1):5–26, 2004.

[31] U. Feige. Relations between average case complexity and approximation complexity. In *34th ACM Symposium on the Theory of Computing*, pages 534–543, 2002.

[32] U. Feige and J. Kilian. On limited versus polynomial nondeterminism. *The Chicago Journal of Theoretical Computer Science*, 1997(1):1–20, 1997.

[33] J. Flum and M. Grohe. **Parameterized Compleixity Theory**. Springer, 2006.

[34] J. Flum, M. Grohe, and M. Weyer. Bounded fixed-parameter tractability and $\log^2 n$ nondeterministic bits. In *31st International Colloquium on Automata, Languages and Programming (ICALP) 2004, Lecture Notes in Computer Science*, volume 3142, pages 555–567. Springer, 2004.

[35] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity, or all languages in np have zero-knowledge proof systems. *Journal of the ACM*, 38(1):691–729, 1991.

[36] J. Goldsmith, M. Levy, and M. Mundhenk. Limited nondeterminism. *SIGACT News*, 27(2):20–29, 1996.

[37] S. Goldwasser and Y. Tauman Kalai. On the (in)security of the Fiat-Shamir paradigm. In *44th IEEE Symposium on Foundations of Computer Science*, pages 102–111, 2003.

[38] I. Haitner, O. Horvitz, J. Katz, C. Koo, R. Morselli, and R. Shaltiel. Reducing complexity assumptions for statistically-hiding commitment. In *Advances in Cryptology – EUROCRYPT ' 2005, Lecture Notes in Computer Science*, volume 3494, pages 58–77. Springer, 2005.

[39] I. Haitner and O. Reingold. Statistically-hiding commitment from any one-way function. In *39th ACM Symposium on the Theory of Computing*, 2007.

[40] D. Harnik and M. Naor. On everlasting security in the *hybrid* bounded storage model. In *33rd International Colloquium on Automata, Languages and Programming (ICALP) 2006, Part II, Lecture Notes in Computer Science*, volume 4052, pages 192–203. Springer, 2006.

[41] D. Harnik and M. Naor. On the compressibility of NP instances and cryptographic applications. In *Electronic Colloquium on Computational Complexity (ECCC), TR06-022*, 2006.

[42] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal of Computing*, 29(4):1364–1396, 1999.

[43] C. Hsiao and L. Reyzin. Finding collisions on a public road, or do secure hash functions need secret coins? In *Advances in Cryptology – CRYPTO '04, Lecture Notes in Computer Science*, volume 3152, pages 92–105. Springer, 2004.

[44] R. Impagliazzo. A personal view of average-case complexity. In *10th Annual Structure in Complexity Theory Conference*, pages 134–147. IEEE Computer Society Press, 1995.

[45] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? In *39th IEEE Symposium on Foundations of Computer Science*, pages 653–663, 1998.

[46] R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *21st ACM Symposium on the Theory of Computing*, pages 44–61, 1989.

[47] Y. Ishai, E. Kushilevitz, and R. Ostrovsky:. Sufficient conditions for collision-resistant hashing. In *2nd Theory of Cryptography Conference – (TCC '05)*, volume 3378 of *Lecture Notes in Computer Science*, pages 445–456, 2005.

[48] H. Kaplan, R. Shamir, and R. Tarjan. Tractability of parameterized completion problems on chordal, strongly chordal, and proper interval graphs. *SIAM Journal of Computing*, 28(5):1906–1922, 1999.

[49] R. Karp. Reducibility among combinatorial problems. *In* **Complexity of Computer Computations***, edited by R. Miller and J. Thatcher, New York: Plenum Press*, pages 85–103, 1972.

[50] R. Karp and M. Rabin. Efficient randomized pattern-matching algorithms. *IBM Journal of Research and Development*, 31(2):249–260, 1987.

[51] J. Kilian. A note on efficient zero-knowledge proofs and arguments. In *24th ACM Symposium on the Theory of Computing*, pages 723–732, 1992.

[52] C. Kintala and P. Fischer. Refining nondeterminism in relativized polynomial-time bounded computations. *SIAM Journal of Computing*, 9(1):46–53, 1980.

[53] M. Li and P. Vitányi. **An Introduction to Kolmogorov Complexity and Its Applications***, 2nd Edition*. Springer Verlag, 1997.

[54] C. Lu. Encryption against space-bounded adversaries from on-line strong extractors. *Journal of Cryptology*, 17(1):27–42, 2004.

[55] U. Maurer. Conditionally-perfect secrecy and a provably-secure randomized cipher. *Journal of Cryptology*, 5(1):53–66, 1992.

[56] U. Maurer. Secret key agreement by public discussion. *IEEE Transaction on Information Theory*, 39(3):733–742, 1993.

[57] U. Maurer, R. Renner, and C. Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In *The 1st Theory of Cryptography Conference – (TCC '04)*, volume 2951 of *Lecture Notes in Computer Science*, pages 21–39, 2004.

[58] S. Micali. CS proofs. In *35th IEEE Symposium on Foundations of Computer Science*, pages 436–453, 1994.

[59] J. Naor and M. Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM Journal on Computing*, 22(4):838–856, 1993.

[60] M. Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991.

[61] M. Naor. On cryptographic assumptions and challenges. In *Advances in Cryptology – CRYPTO '03, Lecture Notes in Computer Science*, volume 2729, pages 96–109. Springer, 2003.

[62] M. Naor, R. Ostrovsky, R. Venkatesan, and M. Yung. Perfect zero-knowledge arguments for NP using any one-way permutation. *Journal of Cryptology*, 11(2):87–108, 1998.

[63] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *21st ACM Symposium on the Theory of Computing*, pages 33–43, 1989.

[64] M. Nguyen, S. Ong, and S. Vadhan. Statistical zero-knowledge arguments for np from any one-way function. In *47th IEEE Symposium on Foundations of Computer Science*, pages 3–14, 2006.

[65] R. Niedermeier. **Invitation to Fixed Parameter Algorithms**. Oxford University Press, 2006.

[66] J.B. Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *Advances in Cryptology – CRYPTO '02, Lecture Notes in Computer Science*, volume 2442, pages 111–126. Springer, 2002.

[67] J.B. Nielsen. On protocol security in the cryptographic model. BRICS Dissertation Series DS-03-8 August, 2003.

[68] C. Papadimitriou. **Computational Complexity**. Addison-Wesley, 1994.

[69] C. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. In *20th ACM Symposium on the Theory of Computing*, pages 229–234, 1988.

[70] C. Papadimitriou and M. Yannakakis. On limited nondeterminism and the complexity of the V-C dimension. *Journal of Computer and System Sciences (JCSS)*, 53(2):161–170, 1996.

[71] D. Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? In *Advances in Cryptology – EUROCRYPT ' 1998, Lecture Notes in Computer Science*, volume 1403, pages 334–345. Springer, 1998.

[72] L. Trevisan, S. Vadhan, and D. Zuckerman. Compression of samplable sources. In *IEEE Conference on Computational Complexity*, pages 1–14, 2004.

[73] S. Vadhan. Constructing locally computable extractors and cryptosystems in the bounded storage model. *Journal of Cryptology*, 17(1):43–77, 2004.

[74] L. Valiant and V. Vazirani. NP is as easy as detecting unique solutions. *Theor. Comput. Sci.*, 47(3):85–93, 1986.

[75] H. Wee. On pseudoentropy versus compressibility. In *IEEE Conference on Computational Complexity*, pages 29–41, 2004.

[76] H. Wee. On obfuscating point functions. In *37th ACM Symposium on Theory of Computing*, pages 523–532, 2005.